

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

• BLACK BORDERS

- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS

• BLACK OR VERY BLACK AND WHITE DARK PHOTOS

- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.**

CoRR. WO 97/10556

(19) 日本国特許庁 (J P)

(12) 公表特許公報 (A)

(11) 特許出願公表番号

特表平11-512543

(43) 公表日 平成11年(1999)10月26日

(51) Int.Cl.⁶

G 0 6 F 17/21
5/00

識別記号

F I

G 0 6 F 15/20
5/00
15/20

5 8 0 L
Z
5 7 0 U

審査請求 未請求 予備審査請求 有 (全 138 頁)

(21) 出願番号 特願平9-512113
(86) (22) 出願日 平成8年(1996)9月13日
(85) 翻訳文提出日 平成10年(1998)3月13日
(86) 国際出願番号 P C T / U S 9 6 / 1 4 6 6 7
(87) 国際公開番号 W O 9 7 / 1 0 5 5 6
(87) 国際公開日 平成9年(1997)3月20日
(31) 優先権主張番号 0 8 / 5 2 7 , 4 3 8
(32) 優先日 1995年9月13日
(33) 優先権主張国 米国 (U S)
(31) 優先権主張番号 0 8 / 5 2 7 , 8 2 7
(32) 優先日 1995年9月13日
(33) 優先権主張国 米国 (U S)

(71) 出願人 アップル コンピュータ インコーポレイ
テッド
アメリカ合衆国 95014 カリフォルニア
州 クバチーノ インフィニット ループ
1-エムエス:38-ビーエイティ
(72) 発明者 エドバーク, ピーター, ケイ.
アメリカ合衆国 97403 オレゴン州 ユ
ージン フェアマウント プールバード
1892
(74) 代理人 弁理士 谷 義一 (外3名)

最終頁に続く

(54) 【発明の名称】 ユニコード・コンバータ

(57) 【要約】

ユニコード・コンバータが開示されている。さらに具体的には、ラウンドトリップ忠実度を備えると共に、その結果の文字コードが他のプラットフォームと互換性をもつことを保証し、文字をソース文字コード化からターゲット文字コードに変換するとき方向および/またはコンテキストを考慮に入れ、変換のために受信したソース文字列が、そのソース文字列が変換対象のソース文字列を格納している入力バッファの長さを越えたときでも別のターゲット文字コード化に正確に変換されることを保証するコード変換および/または切り捨て処理手法が開示されている。このコード変換システムは単一のソース文字列または文字の列を単一のターゲット文字列またはターゲット文字のシーケンスにマッピング (対応づけること) する機能を備えている。ラウンドトリップ忠実度により、ソース・テキストはターゲット・テキストに変換し、再びオリジナル・ソース・テキストに戻るように変換できるので、標準ターゲット文字の使用を最大限にし、プライベート文字の使用を最小限にすることによって互換性を保証している。コード変換は他の文字セット

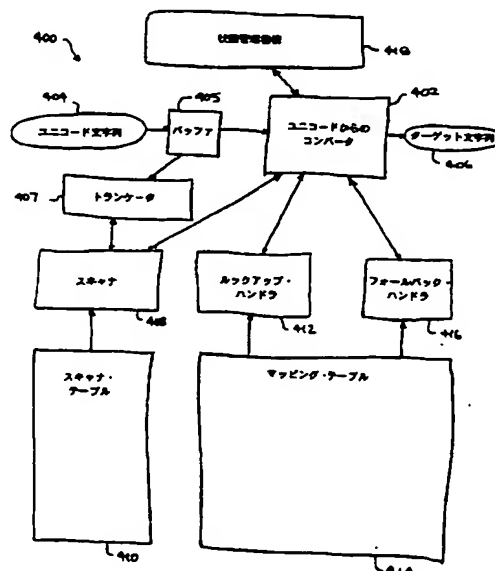


FIG. 4

【特許請求の範囲】

1. ソース文字列をターゲット文字列に変換する方法において、

(a) 第1の文字コード化を有する前記ソース文字列を受け取り、

(b) 前記ソース文字列をテキスト要素に順次分割し、各前記テキスト要素は、前記ソース文字列の文字を1または複数含み、

(c) 各前記テキスト要素の第2の文字コード化に関する変換コードをマッピング・テーブルでルックアップし、

(d) 前記第2の文字コード化の前記ターゲット文字列を形成するように前記テキスト要素の前記変換コードを結合する

ことを特徴とする方法。

2. 請求項1に記載の方法において、前記変換コードは、前記第2の文字コード化における1または複数の文字からなることを特徴とする方法。

3. 請求項1に記載の方法において、前記テキスト要素は、お互いに隣接し、2個以上の文字を含む各前記テキスト要素について、前記文字は前記ソース文字列において隣接することを特徴とする方法。

4. 請求項1に記載の方法において、前記マッピング・テーブルは、レギュラ・マッピングとフォールバック・マッピングを含み、

前記ルックアップ(c)は、前記マッピング・テーブルが、前記テキスト要素についての前記レギュラ・マッピングを使用した変換コードを含まない場合は、前記フォールバック・マッピングを使用して各前記テキスト要素の前記変換コードを決定することを特徴とする方法。

5. 請求項1に記載の方法において、各前記文字は自分に関する文字クラスを有し、

前記分割(b)は、少なくとも部分的には、前記ソース文字列中の前記文字の

前記文字クラスに基づくことを特徴とする方法。

6. 請求項1に記載の方法において、前記分割(b)は、

(b1) 前記ソース文字列から次のソース文字を獲得し、

(b2) 獲得した前記ソース文字を現在のテキスト要素内に含めるべきか、あ

るいはまた、新たな次のテキスト要素を始めるかを決定し、

(b3) 前記決定(b2)に従い、獲得した前記ソース文字を前記現在のテキスト要素内または前記新たな次のテキスト要素内に配置し、

(b4) 前記ソース文字列をテキスト要素内に完全に配置するまで、(b1)から(b3)までを繰り返す

ことを特徴とする方法。

7. 請求項6に記載の方法において、前記決定(b2)は、

(i) 前記ソース文字に関する属性をルックアップし、前記属性は少なくともクラス・インディケータを含み、

(ii) クラス・インディケータに基づき、獲得した前記ソース文字を前記現在のテキスト要素内に含めるべきか、あるいはまた、新たな次のテキスト要素を始めるかを決定する

ことを特徴とする方法。

8. 請求項6に記載の方法において、前記決定(b2)は、

(i) 前記ソース文字に関する属性をルックアップし、前記属性は少なくともクラス・インディケータを含み、

(ii) 複数の状態を有するステート・マシンを提供し、前記ステート・マシンは、前記クラス・インディケータおよび前記ステート・マシンの現在の状態に基づき、獲得した前記ソース文字を前記現在のテキスト要素内に含めるべきか、あるいはまた、新たな次のテキスト要素を始めるかを決定するのに使用され、

(iii) 前記ステート・マシンの前記現在の状態を更新する

ことを特徴とする方法。

9. 請求項1に記載の方法において、前記第1の文字コード化および前記第2の文字コード化の一つは、ユニコード標準に従うことを特徴とする方法。

10. 請求項1に記載の方法において、前記結合(d)は、

(d1) 前記第2の文字コード化に対するターゲット文字サイズを決定し、

(d2) ルックアップされた前記テキスト要素の前記変換コードを、前記ターゲット文字サイズの単位で前記ターゲット文字列にコピーすることにより前記タ

ーゲット文字列を形成する

ことを特徴とする方法。

11. 請求項10に記載の方法において、ルックアップ(c)された前記変換コードは、間接コード化シーケンスまでのオフセットを特定することを特徴とする方法。

12. 請求項1に記載の方法において、さらに、

(e) 前記分割(b)後、前記ルックアップ(c)前に、特定の文字がテキスト要素内に存在する場合は、各前記テキスト要素の前記特定の文字を再配列することを特徴とする方法。

13. 請求項12に記載の方法において、前記再配列は、異なる文字クラスに対する加重値を利用して適切に実行されることを特徴とする方法。

14. ソース文字列をターゲット文字列に変換するコード変換システムにおいて、

第1の文字コード化を有する前記ソース文字列から第2の文字コード化を有する前記ターゲット文字列への変換を制御するコンバータと、

前記コンバータに動作するよう結合され、前記ソース文字列をテキスト要素に分割するスキャナであって、各前記テキスト要素は前記ソース文字列の文字を1

または複数含み、

前記ソース・コード化のテキスト要素に対するターゲット・コード化を記憶するマッピング・テーブルと、

前記コンバータおよび前記マッピング・テーブルに動作するよう結合され、各テキスト要素の第2の文字コード化に関する変換コードをマッピング・テーブルでルックアップするルックアップ・ハンドラと

を備えることを特徴とするシステム。

15. 請求項14に記載のコード変換システムにおいて、さらに、

前記コンバータに動作するよう結合され、前記ルックアップ・ハンドラが1または複数のテキスト要素に対し変換コードを提供できない場合は、フォールバック変換コードを提供するフォールバック・ハンドラを備え、前記フォールバック

変換コードは、前記テキスト要素内の前記文字と完全に等しいわけではないが類似するグラフィカル外観を有する、前記ターゲット・コード化における1または複数のコード・ポイントを含む

ことを特徴とするシステム。

16. 請求項15に記載のコード変換システムにおいて、さらに、

前記入力文字列中の個々の文字を現在のテキスト要素内に含めるべきか、あるいはまた、新たな次のテキスト要素を始めるかの決定について前記スキャナを支援するスキャナ・テーブル手段

を備えることを特徴とするシステム。

17. 請求項14に記載のコード変換システムにおいて、さらに、

前記スキャナに動作するよう結合され、前記入力文字列中の個々の文字を現在のテキスト要素内に含めるべきか、あるいはまた、新たな次のテキスト要素を始めるかの決定について前記スキャナを支援するスキャナ・テーブル

を備えることを特徴とするシステム。

18. 請求項17に記載のコード変換システムにおいて、前記ソース文字列の前記文字は自分に関する文字クラスを有し、

前記スキャナ・テーブルは要素の配列を備え、前記配列は文字クラスによりインデックス付けされていることを特徴とするシステム。

19. 請求項14に記載のコード変換システムにおいて、前記ソース文字列中の前記文字は、ユニコード文字であることを特徴とするシステム。

20. 請求項14に記載のコード変換システムにおいて、前記ターゲット文字列中の前記文字は、ユニコード文字であることを特徴とするシステム。

21. ソース文字列をターゲット文字列に変換するコード変換システムにおいて、

第1の文字コード化を有する前記入力文字列から第2の文字コード化を有する前記ターゲット文字列への変換を制御するコンバータ手段と、

前記ソース文字列をテキスト要素に分割するステート・マシンであって、各テキスト要素は前記ソース文字列の文字を1または複数含むステート・マシン手段

と、

前記ソース・コード化のテキスト要素に対するターゲット・コード化を記憶するマッピング手段と、

各前記テキスト要素の第2の文字コード化に関する変換コードを前記マッピング手段でルックアップするルックアップ・ハンドラ手段と

を備えることを特徴とするシステム。

22. 請求項21に記載のコード変換システムにおいて、さらに、

前記入力文字列中の個々の文字を現在のテキスト要素内に含めるべきか、あるいはまた、新たな次のテキスト要素を始めるかの決定について前記スキャナを支援するスキャナ・テーブル手段

を備えることを特徴とするシステム。

23. 請求項22に記載のコード変換システムにおいて、さらに、

前記ルックアップ・ハンドラ手段が1または複数のテキスト要素に対し変換コードを提供できない場合は、フォールバック変換コードを提供するフォールドバック・ハンドラ手段を備え、前記フォールバック変換コードは、前記テキスト要素内の文字と完全に等しいわけではないが、類似するグラフィカル外観を有する、前記ターゲット・コード化における1または複数のコード・ポイントを含むことを特徴とするシステム。

24. 入力文字列をスキャンするスキャンング・システムにおいて、

前記入力文字列から入力文字を獲得する入力装置であって、前記入力文字列は文字コード化を有し、前記入力文字列の各文字は文字クラスを有する入力装置と

前記入力文字の属性を提供する属性テーブルと、

前記入力文字の前記属性および前記ステート・マシンの現在の状態に従って、前記ステート・マシンの次の状態および次のアクションの両方を決定するステート・マシンとを備え、

前記ステート・マシンが決定した前記次のアクションに基づいて、前記スキャンング・システムは、前記入力文字列の前記入力文字を現在のテキスト要素内に

含めるべきか、あるいはまた、前記現在のテキスト要素を終え新たなテキスト要素を始めるかを決定する

ことを特徴とするシステム。

25. 請求項24に記載のスキャニング・システムにおいて、前記属性は少なくとも前記入力文字の文字クラスを含み、

前記ステート・マシンは、前記入力文字の前記文字クラスおよび前記ステート・マシンの前記現在の状態に従って、前記次の状態および前記次のアクションを決定することを特徴とするシステム。

26. ソース文字列をターゲット文字列に変換するプログラム命令を含むコンピュータ読み取り可能な媒体において、

第1の文字コード化を有するソース文字列をコンピュータが受け取るように構成されたコンピュータ読み取り可能なコードと、

前記ソース文字列をテキスト要素にコンピュータが分割するように構成され、各テキスト要素は前記ソース文字列の文字を1または複数含むコンピュータ読み取り可能なコードと、

各前記テキスト要素に対する第2の文字コード化に関する変換コードをコンピュータがルックアップするように構成されたコンピュータ読み取り可能なコードと、

前記第2の文字コード化の前記ターゲット文字列を形成するように前記テキスト要素の前記変換コードをコンピュータが結合するように構成されたコンピュータ読み取り可能なコードと

を備えることを特徴とするコンピュータ読み取り可能な媒体。

27. ソース文字列をターゲット文字列に変換する方法において、

(a) 第1の文字コード化を有するソース文字列を受け取り、前記ソース文字列は複数のソース文字を含み、

(b) 前記ソース文字列の前記ソース文字の方向を決定し、

(c) 前記第1の文字コード化および決定された前記方向に基づき、各前記ソース文字に対する第2の文字コード化に関する変換コードをマッピング・テーブル

ルでルックアップし、

(d) 前記第2の文字コード化のターゲット文字列を形成するように、前記ソース文字の前記変換コードを結合する

ことを特徴とする方法。

28. 請求項27に記載の方法において、各ソース文字の前記方向は、左から右への方向および右から左への方向の一つであることを特徴とする方法。

29. 請求項27に記載の方法において、前記決定(b)は、

(b1) 前記方向が不適切かどうかを決定し、

(b2) 前記方向が適切な場合は、方向が、左から右への方向および右から左への方向の一つを決定する

ことを備えることを特徴とする方法。

30. 請求項27に記載の方法において、前記決定(b)は、前記ソース文字列の1または複数の文字の前記方向に基づき、前記ソース文字列のすべてまたは一部の前記方向を決定することを特徴とする方法。

31. 請求項27に記載の方法において、前記決定(b)は、

(i) 前記ソース文字に関する属性をルックアップし、前記属性は少なくともクラス・インディケータを含み、

(ii) 前記クラス・インディケータに基づき前記ソース文字の前記方向を決定する

ことを備えることを特徴とする方法。

32. 請求項27に記載の方法において、前記決定(b)は、

(i) 前記ソース文字に関する属性をルックアップし、前記属性は少なくともクラス・インディケータを含み、

(ii) 複数の状態を有するステート・マシンを提供し、前記ステート・マシンは、前記クラス・インディケータおよび前記ステート・マシンの状態に基づき、前記ソース文字の方向を決定するために使用され、

(iii) 前記ステート・マシンの状態を更新する

ことを備えることを特徴とする方法。

33. 請求項32に記載の方法において、前記ステート・マシンは、さらに、前記クラス・インディケータおよび前記ステート・マシンの現在の状態に基づいて、前記ソース文字を現在のテキスト要素内に含めるべきか、あるいはまた、新

たな次のテキスト要素を始めるかを決定することを特徴とする方法。

34. 請求項32に記載の方法において、前記決定(b)は、前記現在のテキスト要素の前記方向を決定することを特徴とする方法。

35. 請求項27に記載の方法において、さらに、(e)前記ソース文字列をテキスト要素に順次分割し、各テキスト要素は前記ソース文字列の文字を1または複数含むことを備え、

前記決定(b)は、前記テキスト要素の前記方向を決定し、

前記ルックアップ(c)は、前記テキスト要素の前記1または複数のソース文字の前記第1の文字コード化および前記テキスト要素の前記方向に基づき、各前記テキスト要素の第2の文字コード化に関する変換コードを前記マッピング・テーブルでルックアップすることを特徴とする方法。

36. 請求項35に記載の方法において、前記マッピング・テーブルは、レギュラ・マッピングとフォールバック・マッピングを含み、

前記ルックアップ(c)は、前記マッピング・テーブルが、前記レギュラ・マッピングを使用する前記テキスト要素に対する変換コードを含まない場合は、前記フォールドバック・マッピングを使用して前記テキスト要素の前記変換コードを決定することを特徴とする方法。

37. 請求項35に記載の方法において、各前記ソース文字は、自分に関する文字クラスを有し、

前記分割(e)は、少なくとも部分的には、前記ソース文字列中の前記ソース文字の前記文字クラスに基づくことを特徴とする方法。

38. 請求項27に記載の方法において、前記第1の文字コード化および前記第2の文字コード化の一つは、ユニコード標準に従うことを特徴とする方法。

39. ソース文字列をターゲット文字列に変換するコード変換システムにおいて

ソース文字コード化を有する入力文字列からターゲットの文字コード化を有するターゲット文字列への前記変換を制御するコンバータであって、前記入力文字列は複数の文字を含むコンバータと、

前記コンバータに動作するよう結合され、前記入力文字列中の前記文字の方向を決定するスキャナと、

前記ソース・コード化の文字に対するターゲット・コード化を記憶するマッピング・テーブルと、

前記コンバータおよび前記マッピング・テーブルに動作するよう結合され、前記方向および前記入力文字列中の前記文字に対する前記ソース・コード化に基づき、前記入力文字列中の各前記文字についてのターゲット文字コード化に関する変換コードを前記マッピング・テーブルでルックアップするルックアップ・ハンドラと

を備えることを特徴とするシステム。

40. 請求項39に記載のコード変換システムにおいて、

前記スキャナは、さらに、前記入力文字列をテキスト要素に分割し、各前記テキスト要素は前記入力文字列の文字を1または複数含み、

前記スキャナは、前記テキスト要素の前記方向を決定し、

前記マッピング・テーブルは、前記ソース・コード化のテキスト要素に対するターゲット・コード化を記憶し、

前記ルックアップ・ハンドラは、前記方向および前記テキスト要素内の前記文字についての前記ソース・コード化に基づき、各前記テキスト要素に対する前記ターゲット文字コード化をルックアップすることを特徴とするシステム。

41. 請求項40に記載のコード変換システムにおいて、さらに、

前記コンバータに動作するよう結合され、前記ルックアップ・ハンドラが1または複数のテキスト要素に対し変換コードを提供できない場合は、フォールバック

変換コードを提供するフォールバック・ハンドラを備え、前記フォールバック変換コードは、前記テキスト要素内の文字と完全に等しいわけではないが、類似

するグラフィカル外観を有する、前記ターゲット・コード化における1または複数のコード・ポイントを含む

ことを特徴とするシステム。

42. 請求項41に記載のコード変換システムにおいて、さらに、

前記入力文字列の前記文字の前記方向の決定、および、前記入力文字列中の個々の文字を現在のテキスト要素内に含めるべきか、あるいはまた、新たな次のテキスト要素を始めるかの決定について前記スキャナを支援するスキャナ・テーブル手段

を備えることを特徴とするシステム。

43. 請求項39に記載のコード変換システムにおいて、前記入力文字列の前記文字は、自分に関する文字クラスを有し、

前記スキャナ・テーブルは、要素の配列を備え、前記配列は、文字クラスによりインデックス付けされていることを特徴とするシステム。

44. 請求項39に記載のコード変換システムにおいて、前記ソース文字列中の前記文字は、ユニコード文字であることを特徴とするシステム。

45. 請求項39に記載のコード変換システムにおいて、前記ターゲット文字列中の前記文字は、ユニコード文字であることを特徴とするシステム。

46. ソース文字列をターゲット文字列に変換するコード変換システムにおいて

ソース文字コード化を有する前記ソース文字列からターゲット文字コード化を有するターゲット文字列への前記変換を制御するコンバータ手段と、

前記ソース文字列中の各前記文字の方向を決定するステート・マシン手段

と、

前記ソース文字コード化の文字に対するターゲット文字コード化を記憶するマッピング手段と、

前記決定された方向を考慮しつつ、前記ソース文字コード化に従い、前記ソース文字列中の各前記文字に対するターゲット文字コード化に関する変換コードを前記マッピング手段でルックアップするルックアップ・ハンドラ手段と

を備えることを特徴とするシステム。

47. 請求項46に記載のコード変換システムにおいて、前記ステート・マシン手段は、さらに、前記ソース文字列をテキスト要素に分割し、各前記テキスト要素は、前記ソース文字列の文字を1または複数含むことを特徴とするシステム。

48. 請求項47に記載のコード変換システムにおいて、さらに、

前記ルックアップ・ハンドラ手段が1または複数のテキスト要素に対し変換コードを提供できない場合は、フォールバック変換コードを提供するフォールバック・ハンドラ手段を備え、前記フォールバック変換コードは、前記テキスト要素内の前記文字と完全に等しいわけではないが、類似するグラフィカル外観を有する、前記ターゲット・コード化における1または複数のコード・ポイントを含むことを特徴とするシステム。

49. ソース文字列をターゲット文字列に変換するプログラム命令を含むコンピュータ読み取り可能な媒体において、

第1の文字コード化を有するソース文字列をコンピュータが受け取るように構成されたコンピュータ読み取り可能なコードと、

前記ソース文字列中の各前記ソース文字の方向をコンピュータが決定し、前記ソース文字列をテキスト要素にコンピュータが分割するように構成され、各前記テキスト要素はソース文字列の文字を1または複数含むコンピュータ読み取り可能なコードと、

各前記テキスト要素の第2の文字コード化に関する変換コードをコンピュータがルックアップするように構成されたコンピュータ読み取り可能なコードと、

前記第2の文字コード化のターゲット文字列を形成するように前記テキスト要素の変換コードをコンピュータが結合するように構成されたコンピュータ読み取り可能なコードと

を備えることを特徴とするコンピュータ読み取り可能な媒体。

50. ソース文字列をターゲット文字列に変換する方法において、

(a) 第1の文字コード化を有するソース文字列を受け取り、前記ソース文字列は複数のソース文字を含み、

(b) 前記ソース文字列中の各前記ソース文字のコンテキストを決定し、

(c) 前記第1の文字コード化および各前記ソース文字の決定された前記コンテキストに基づき、各前記ソース文字の第2の文字コード化に関する変換コードをマッピング・テーブルでルックアップし、

(d) 前記第2の文字コード化のターゲット文字列を形成するように前記ソース文字の前記変換コードを結合する

ことを備えることを特徴とする方法。

51. 請求項50に記載の方法において、各ソース文字の前記コンテキストは、前記ソース文字列中の自分と隣接するソース文字に依存することを特徴とする方法。

52. 請求項50に記載の方法において、前記決定(b)は、

(b1) コンテキストが不適切かどうかを決定し、

(b2) コンテキストが適切な場合は、コンテキストが、初期、中間、終了またはアローンの一つであるかどうかを決定する

ことを備えることを特徴とする方法。

53. 請求項52に記載の方法において、前記決定(b2)を実行する際に、特

別なソース文字の前記コンテキストの決定に関し、隣接のソース文字が前記特別なソース文字のコンテキストに対し影響するように、前記ソース文字列は一文字ずつスキャンされることを特徴とする方法。

54. 請求項50に記載の方法において、前記決定(b)は、

(i) 前記ソース文字に関する属性をルックアップし、前記属性は少なくともクラス・インディケータを含み、

(ii) 前記クラス・インディケータに基づき、前記ソース文字の前記コンテキストを決定する

ことを備えることを特徴とする方法。

55. 請求項50に記載の方法において、前記決定(b)は、

(i) 前記ソース文字に関する属性をルックアップし、前記属性は少なくともクラス・インディケータを含み、

(ii) 複数の状態を有するステート・マシンを提供し、前記ステート・マシンは前記クラス・インディケータおよび前記ステート・マシンの状態に基づき、前記ソース文字の前記コンテキストを決定するために利用され、

(iii) 前記ステート・マシンの状態を更新する

ことを備えることを特徴とする方法。

56. 請求項55に記載の方法において、前記ステート・マシンは、さらに、前記クラス・インディケータおよび前記ステート・マシンの現在の状態に基づいて、前記ソース文字を現在のテキスト要素内に含めるべきか、あるいはまた、新たな次のテキスト要素を始めるかを決定することを特徴とする方法。

57. 請求項50に記載の方法において、さらに、

(e) 前記ソース文字列をテキスト要素に順次分割し、各テキスト要素は前記ソース文字列の文字を1または複数含むことを備え、

前記ルックアップ(c)は、各前記テキスト要素の前記第2の文字コード化に

関する変換コードを前記マッピング・テーブルでルックアップし、ルックアップされた変換コードの1つを選択し、前記結合(d)は、前記第2の文字コード化による前記ターゲット文字列を形成するように前記テキスト要素に対する前記変換コードを結合することを特徴とする方法。

58. 請求項57に記載の方法において、前記マッピング・テーブルは、レギュラ・マッピングとフォールバック・マッピングを含み、

前記ルックアップ(c)は、前記マッピング・テーブルが、前記レギュラ・マッピングを使用する前記テキスト要素に対する変換コードを含まない場合は、前記フォールバック・マッピングを使用して各前記テキスト要素の前記変換コードを決定することを特徴とする方法。

59. 請求項57に記載の方法において、各前記ソース文字は、自分に関する文字クラスを有し、

前記分割(e)は、少なくとも部分的には、前記ソース文字列中の前記ソース文字の前記文字クラスに基づくことを特徴とする方法。

60. 請求項57に記載の方法において、前記分割(e)は、

(e 1) 前記ソース文字列から次のソース文字を獲得し、

(e 2) 獲得した前記ソース文字を現在のテキスト要素内に含めるべきか、あるいはまた、新たな次のテキスト要素を始めるかを決定し、

(e 3) 前記決定(e 2)に従い、獲得した前記ソース文字を前記現在のテキスト要素内または前記新たな次のテキスト要素内に配置し、

(e 4) 前記ソース文字列をテキスト要素内に完全に配置するまで、(e 1)から(e 3)までを繰り返す

ことを備えることを特徴とする方法。

61. 請求項60に記載の方法において、前記決定(e 2)は、

(i) 前記ソース文字に関する属性をルックアップし、前記属性は少なくともクラス・インディケータを含み、

(ii) 複数の状態を有するステート・マシンを提供し、前記ステート・マシンは、前記クラス・インディケータおよび前記ステート・マシンの現在の状態に基づき、獲得した前記ソース文字を前記現在のテキスト要素内に含めるべきか、あるいはまた、新たな次のテキスト要素を始めるかを決定するのに利用され、

(iii) 前記ステート・マシンの前記現在の状態を更新する

ことを備えることを特徴とする方法。

62. 請求項57に記載の方法において、前記第1の文字コード化および前記第2の文字コード化の一つは、ユニコード標準に従うことを特徴とする方法。

63. 請求項57に記載の方法において、前記結合(d)は、

(d 1) 前記第2の文字コード化のターゲット文字サイズを決定し、

(d 2) ルックアップされた前記テキスト要素の前記変換コードを前記ターゲット文字サイズの単位で前記ターゲット文字列にコピーすることにより前記ターゲット文字列を形成する

ことを備えることを特徴とする方法。

64. ソース文字列をターゲット文字列に変換するコード変換システムにおいて

ソース文字コード化を有する前記入力文字列からターゲット文字コード化を有

する前記ターゲット文字列への前記変換を制御するコンバータであって、前記入力文字列は複数の文字を含むコンバータと、

前記コンバータに動作するよう結合され、前記入力文字列中の各前記文字のコンテキストを決定するスキャナと、

前記ソース・コード化の文字に対するターゲット・コード化を記憶するマッピング・テーブルと、

前記コンバータおよび前記マッピング・テーブルに動作するよう結合され、前記コンテキストおよび前記入力文字列中の前記文字に対する前記ソース・コード

化に基づき、前記入力文字列中の各前記文字に対するターゲット文字コード化に関する変換コードを前記マッピング・テーブルでルックアップするルックアップ・ハンドラと

を備えることを特徴とするシステム。

65. 請求項64に記載の方法において、

前記スキャナは、さらに、前記入力文字列を前記テキスト要素に分割し、各テキスト要素は前記入力文字列の文字を1または複数含み、

前記マッピング・テーブルは、前記ソース・コード化のテキスト要素に対するターゲット・コード化を記憶し、

前記ルックアップ・ハンドラは、前記コンテキストおよび前記テキスト要素内の前記文字に対する前記ソース・コード化に基づき、各テキスト要素に対する前記ターゲット・文字コード化を選択することを特徴とする方法。

66. 請求項65に記載のコード変換システムにおいて、さらに、

前記コンバータに動作するよう結合され、前記ルックアップ・ハンドラが1または複数のテキスト要素に対し変換コードを提供できない場合は、フォールバック変換コードを提供するフォールバック・ハンドラを備え、前記フォールバック変換コードは、前記テキスト要素内の文字と完全に等しいわけではないが、類似するグラフィカル外観を有する、ターゲット・コード化における1または複数のコード・ポイントを含むことを特徴とするシステム。

67. 請求項66に記載のコード変換システムにおいて、さらに、

前記入力文字列中の個々の文字を現在のテキスト要素内に含めるべきか、あるいはまた新たな次のテキスト要素を始めるかの決定について前記スキャナを支援するスキャナ・テーブル手段

を備えることを特徴とするシステム。

68. 請求項64に記載のコード変換システムにおいて、前記入力文字列の前記文字は、自分に関する文字クラスを有し、

前記スキャナ・テーブルは、要素の配列を備え、前記配列は、前記文字クラスによりインデックス付けされていることを特徴とするシステム。

69. 請求項64に記載のコード変換システムにおいて、前記ソース文字列中の前記文字は、ユニコード文字であることを特徴とするシステム。

70. 請求項64に記載のコード変換システムにおいて、前記ターゲット文字列中の前記文字は、ユニコード文字であることを特徴とするシステム。

71. ソース文字列をターゲット文字列に変換するコード変換システムにおいて

、
ソース文字コード化を有する前記ソース文字列からターゲット文字コード化を有する前記ターゲット文字列への前記変換を制御するコンバータ手段と、

前記ソース中の各前記文字のコンテキストを決定するステート・マシン手段と

、
前記ソース文字コード化を有する文字に対するターゲット文字コード化を記憶するマッピング手段と、

前記ソース文字列中の各前記文字についてのターゲット文字コード化に関する変換コードを前記マッピング手段でルックアップするルックアップ・ハンドラ手段と

を備えることを特徴とするシステム。

72. 請求項71に記載のコード変換システムにおいて、前記ステート・マシン手段は、さらに、前記ソース文字列をテキスト要素に分割し、各テキスト要素は、前記ソース文字列の文字を1または複数含むことを特徴とするシステム。

73. 請求項72に記載のコード変換システムにおいて、さらに、

前記ルックアップ・ハンドラ手段が1または複数のテキスト要素に対し変換コードを提供できない場合は、フォールバック変換コードを提供するフォールバック・ハンドラ手段を備え、前記フォールバック変換コードは、前記テキスト要素内の前記文字と完全に等しいわけではないが、類似するグラフィカル外観を有する、前記ターゲット・コード化における1または複数のコード・ポイントを含むことを特徴とするシステム。

74. ソース文字列をターゲット文字列に変換するプログラム命令を含むコンピュータ読み取り可能な媒体において、

第1の文字コード化がなされた前記ソース文字列をコンピュータが受け取るように構成されたコンピュータ読み取り可能なコードと、

前記ソース文字列中の各前記ソース文字のコンテキストをコンピュータが決定し、前記ソース文字列をテキスト要素にコンピュータが分割するように構成され、各テキスト要素は前記ソース文字列の文字を1または複数含むコンピュータ読み取り可能なコードと、

各前記テキスト要素の第2の文字コード化に関する変換コードをコンピュータがルックアップするように構成されたコンピュータ読み取り可能なコードと、

前記第2の文字コード化によるターゲット文字列を形成するように前記テキスト要素の前記変換コードをコンピュータが結合するように構成されたコンピュータ読み取り可能なコードと

を備えることを特徴とするコンピュータ読み取り可能な媒体。

75. ソース文字列をターゲット文字列に変換するコード変換システムにおいて、

第1の文字コード化を有する前記ソース文字列から第2の文字コード化を有する前記ターゲット文字列への前記変換を制御するコンバータと、

前記ソース文字列の部分の1つを一度に受け取るためのバッファであって、前記ソース文字列は部分を2つ以上含むバッファと、

前記ソース文字列の前記部分を切り捨てるランケータと、

前記コンバータに動作するよう結合され、前記ソース文字列の切り捨てられた

部分をテキスト要素に分割するスキャナであって、各テキスト要素は、前記ソース文字列の前記切り捨てられた部分の文字を1または複数含むスキャナと、

前記ソース・コード化の前記テキスト要素に対するターゲット・コード化を記憶するマッピング・テーブルと、

前記コンバータおよび前記マッピング・テーブルに動作するよう結合され、各前記テキスト要素の第2の文字コード化に関する変換コードを前記マッピング・テーブルでルックアップするルックアップ・ハンドラと

を備えることを特徴とするシステム。

76. 請求項75に記載のコード変換システムにおいて、前記ランケータは、前記スキャナとともに、前記ソース文字列の前記部分を通してスキャンして前記部分内の前記テキスト要素を決定し、前記切り捨てられた部分を、完全なテキスト要素からなる前記ソース文字列の部分のサブパートとして決定することを特徴とするシステム。

77. 請求項75に記載のコード変換システムにおいて、前記ランケータは、前記切り捨てられた部分を、前記ソース文字列の後続の部分に影響を受けることなく前記ターゲット文字列に変換できる、前記ソース文字列の部分のサブパートとして決定することを特徴とするシステム。

78. 請求項75に記載のコード変換システムにおいて、前記ソース文字列はテキスト文字列であり、前記ターゲット文字列はテキスト文字列であることを特徴とするシステム。

79. 請求項75に記載のコード変換システムにおいて、さらに、

前記コンバータに操作的に接続され、前記ルックアップ・ハンドラが1または複数のテキスト要素に対し変換コードを提供できない場合は、フォールバック変

換コードを提供するフォールバック・ハンドラを備え、前記フォールバック変換コードは、前記テキスト要素内の前記文字と完全に等しいわけではないが、類似するグラフィカル外観を有する、前記ターゲット・コード化における1または複数のコード・ポイントを含む

ことを特徴とするシステム。

80. 請求項75に記載のコード変換システムにおいて、さらに、

前記入力文字列中の個々の文字を現在のテキスト要素内に含めるべきか、あるいはまた、新たな次のテキスト要素を始めるかの決定について前記スキャナを支援するスキャナ・テーブル手段

を備えることを特徴とするシステム。

81. 請求項75に記載のコード変換システムにおいて、さらに、

前記スキャナに動作するよう結合され、前記入力文字列中の個々の文字を現在のテキスト要素内に含めるべきか、あるいはまた、新たな次のテキスト要素を始めるかの決定について前記スキャナを支援するスキャナ・テーブル

を備えることを特徴とするシステム。

82. 請求項81に記載のコード変換システムにおいて、前記ソース文字列の前記文字は自分に関する文字クラスを有し、

前記スキャナ・テーブルは要素の配列を備え、前記配列は文字クラスによりインデックス付けされていることを特徴とするシステム。

83. 請求項75に記載のコード変換システムにおいて、前記ソース文字列中の前記文字は、ユニコード文字であることを特徴とするシステム。

84. 請求項75に記載のコード変換システムにおいて、前記ターゲット文字列中の前記文字は、ユニコード文字であることを特徴とするシステム。

85. ターゲット文字列への文字変換のためにソース文字列を切り捨てる方法において、

(a) バッファ内のソース文字列のバッファ部分を受け取り、前記ソース文字列は2個以上のバッファ部分を含み、

(b) 後続の前記ソース文字列のバッファ部分に影響を受けることなくターゲット文字列に変換できる、前記ソース文字列の前記バッファ部分のサブパートを決定し、

(c) 前記ソース文字列の前記バッファ部分の前記サブパートをターゲット・コード化に変換する

ことを特徴とする方法。

86. 請求項85に記載の方法において、さらに、

(d) 次のバッファ部分またはそのサブパートとともに変換するために前記バッファ部分の残り部分をセーブし、前記残り部分はサブパートとともに前記バッファ部分と等しい

ことを特徴とする方法。

87. 請求項85に記載の方法において、前記ソース文字列はテキスト文字列であり、前記ターゲット文字列はテキスト文字列であることを特徴とする方法。

88. ターゲット文字列への文字変換のためにソース文字列を切り捨てる方法において、

(a) バッファ内のソース文字列の部分を受け取り、

(b) 前記ソース文字列の前記部分内のテキスト要素を決定し、各前記テキスト要素は前記ソース文字列の文字を1または複数含み、

(c) 前記テキスト要素が完全かどうかを決定し、

(d) 前記テキスト要素が完全であれば、前記テキスト要素を前記ソース文字列の切り捨てられた部分内に含め、

(e) 前記ソース文字列の前記部分が完全に考慮されるまで、ステップ(b)

～(d)を繰り返し、

(f) その後、文字変換のために前記ソース文字列の前記切り捨てられた部分を出力する

ことを特徴とする方法。

89. 請求項88に記載の方法において、さらに、

(g) 前記バッファ内に受け取られた前記ソース文字列の次の部分とともに使用するために、前記ソース文字列のすべての残りの部分をセーブする

ことを特徴とする方法。

90. 請求項89に記載の方法において、前記ソース文字列はテキスト文字列であり、前記ターゲット文字列はテキスト文字列であることを特徴とする方法。

91. ソース文字列をターゲット文字列に変換するプログラム命令を含むコンピュータ読み取り可能な媒体において、

第1の文字コード化がなされた前記ソース文字列をバッファにコンピュータが受け取るように構成されたコンピュータ読み取り可能なコードと、

ソース文字列の部分をコンピュータが切り捨てるように構成されたコンピュータ読み取り可能なコードと、

前記ソース文字列の切り捨てられた部分をテキスト要素にコンピュータが分割するように構成され、各前記テキスト要素は前記ソース文字列の切り捨てられた部分の文字を1または複数含むコンピュータ読み取り可能なコードと、

各前記テキスト要素の第2の文字コード化に関する変換コードをコンピュータがルックアップするように構成されたコンピュータ読み取り可能なコードと、

前記第2の文字コード化によるターゲット文字列を形成するように前記テキスト要素の前記変換コードをコンピュータが結合するように構成されたコンピュータ読み取り可能なコードと

を備えることを特徴とするコンピュータ読み取り可能な媒体。

92. 請求項91に記載のコンピュータ読み取り可能な媒体において、ソース文字列の部分をコンピュータが切り捨てるように構成されたコンピュータ読み取り可能なコードは、前記切り捨てられた部分を、後続のソース文字列の前記部分に影響を受けることなく前記ターゲット文字列に変換できる、前記ソース文字列の部分のサブパートとして決定することを特徴とするコンピュータ読み取り可能な媒体。

【発明の詳細な説明】

ユニコード・コンバータ

説明技術分野

本発明は記述または表示されるテキストの文字コード間で変換を行うシステムに関し、さらに具体的には、ある文字セットと別の文字セット間で変換を行うコード・コンバータに関する。

背景技術

コンピュータや他のエレクトロニック・デバイスはテキストを使用してユーザと対話するのが一般的である。テキストはモニタや他のタイプの表示デバイスから表示されるのが普通である。テキストはコンピュータや他のエレクトロニック・デバイスの内部ではデジタル形式で表現しなければならないので、文字セット・コード化を使用しなければならない。一般的には、文字セット・コード化は文字セットの各文字を固有のデジタル表現でコード化するように働く。文字（コード化される）は文字、数字および種々のテキスト・シンボルに対応し、コンピュータや他のエレクトロニック・デバイスで使用される数値コードが割り当てられている。コンピュータや他のエレクトロニック・デバイスで使用されている最もよく知られた文字セットは情報交換用米国標準コード(American National Standard Code for Information Exchange - A S C I I)である。A S C I Iはそのコード化のために7ビット・シーケンスを使用している。他の国では、異なる文字セットが使用されている。ヨーロッパでは、支配的な文字コード化標準は国際標準化機構(International Standards Organization - I S O)によって開発されたI S O 8859-X、特にI S O 8859-1(「L a t i n - 1」と呼ばれている)である。日本では、支配的な文字コード化標準はJ I S X 0208であり、ここでJ I Sとは日本情報標準(Japanese Information Standard)のことであり、日本標準協会(Japan Standards

Association - J S A)によって開発されたものである。その他の既存文字セットの例としては、M a c (登録商標) O S S t a n d a r d R o m a n コー

ド化 (Apple Computer, Inc 開発)、Shift-JIS (日本)、Big 5 (台湾)、その他多くがある。

ビジネスとネットワークのグローバル化に伴い、コンピュータや他のエレクトロニック・デバイスが複数の文字コード化を処理する能力をもつことが重要になっている。例えば、同じコンピュータやエレクトロニック・デバイスは、コンピュータや他のエレクトロニック・デバイスと異なる言語で対話することを望んでいる異なる国籍の人達で利用されることがある。このような各言語では、異なる文字セット・コード化が必要になるのが通常である。しかし、文字セットは同じ言語の場合でも、異なることがある。

もう1つの要求は、ある文字セット・コードから別の文字セット・コード化への変換ができることである。例えば、ISO 8859-1を使用するフランスのユーザは、電子メール・メッセージをISO 8859-8を使用するイスラエルのユーザにフランス語で送りたい場合がある。送信者と受信者は異なる文字セット・コード化を使用しているので、メッセージの中の非ASCII文字は、イスラエルのユーザには誤って伝えられることになる。理想的なことは、コンピュータまたはエレクトロニック・デバイスの1つがある文字セットから別の文字セットに変換することである。このことは、限られた範囲でいくつかの文字セット間ですでに達成されているが、最新のコンピュータやエレクトロニック・デバイスでもほとんど不可能である。コード変換を困難にしているのは、異なる文字標準が多数存在し、各国の標準が矛盾し、または不統一であることがよくあるためである。

Unicode (登録商標) 標準 (以下では、単にユニコードまたはユニコード標準という) は国際的文字コード化標準となるために開発されたものである。ユニコード標準の設計者はより効率的で、柔軟性のある文字識別方法を必要として、提供したものである。ユニコード標準には、1990年12月31以前に承認され、公表されたすべての主要国際標準のすべての文字が含まれている他に、以前の標準になかった他の文字も含まれている。これらの文字は重複することなくユニコード標準にコード化されている。ユニコード標準に含まれるコードは1

6 ビット（または2 バイト）幅である。

ユニコード標準などの文字セット標準はコード変換を容易にし、テキスト・データに働きかける有用なプロセスのインプリメンテーションを可能にしている。例えば、上記の例によれば、フランス国内のコンピュータや他のエレクトロニック・デバイスはユニコード文字を送信することができ、イスラエル国内のコンピュータや他のエレクトロニック・デバイスは受信したユニコード文字を、イスラエル国内のコンピュータや他のエレクトロニック・デバイスと互換性のあるヘブライ語の文字に変換することができる。

以下では、ユニコード標準の概要について説明する。なお、ユニコード標準の詳細は、例えば、「The Unicode Standard, Worldwide Character Encoding, Version 1.0, Addison-Wesley 1991」に記載されている（バージョン1.1にも詳細が記載されている）。これらのどちらのバージョンも、その全体は引用により本明細書の一部を構成するものである。

ユニコード・コード化方式の設計は方向性を除き、基本的テキスト処理アルゴリズムの設計から独立している。ユニコード・インプリメンテーションは適当なテキスト処理および／またはレンダリング・アルゴリズムを含んでいるものと想定されている。便宜上、ユニコード標準のすべてのコードは言語および機能のカテゴリ別に分類されているが、ユニコード標準のすべてのコードは等しくアクセス可能になっている。「ユニコード標準のコード・スペースは6つのゾーンに分割されている。すなわち、一般的スクリプト（アルファベットおよび相対的に小さな文字セットをもつ他のスクリプト）、シンボル（記号）、C J K（中国語、日本語および朝鮮語）補足文字、C J K表意文字、プライベート使用および互換性である。一般的スクリプト・ゾーンはラテン、キリル、ギリシア、ヘブライ、アラビア、デーヴァナーガリーおよびタイなどのアルファベットまたは音節スクリプトを含んでいる。シンボル・ゾーンは句読点、数学、化学、読者の注意をひく記号などの、様々な種類の文字を含んでいる。C J K補足ゾーンは句読点、シンボル、カナ文字、B o p o m o f o 文字、および単一および複合ハングル文字を含んでいる。C J K表意文字ゾーンには、20,000個以上の表意文字また

は他のスクリプトからの文字のためのスペースが用意されている。プライベート使用ゾーンはユーザまたはベンダ固有のグラフィック文字を定義するために使用される。互換性ゾーンはユニコード・コード化で他の規範的表現をもつ、幅広く使用されている企業および国内標準からの文字を含んでいる。」（上記標準バージョン1.0の13ページから引用）。

ユニコード標準は文字特性と制御文字を含んでいる。文字特性はユニコード・コード化に含まれるコード・ポイントに関する意味論的知識を必要とする解析、ソート、および他のアルゴリズムで利用すると便利である。ユニコード標準で指定されている文字特性には、ディジット、数字、スペース文字、非スペース・マーク、および方向がある。ユニコード文字は文字特性に基づいてグループ化されている。ディジット、数字およびスペース文字は周知である。非スペース・マーク・グループは非スペース・マークを収容し、方向グループは方向文字を収容している。

非スペース・マーク（例えば、ギリシアとローマ・スクリプトにおけるアクセント・マーク、アラビアとデーヴァナーガリーにおける母音マーク）は最終的にレンダリングされたテキストには線形的に現れない。ユニコード文字コード・シーケンスでは、このような文字はすべて、その文字で修正されたベース文字、またはその後に音声順に分節される文字のあとに置かれている（例えば、ローマ文字の“A”は事前構成形式でストアされないとき“A'”としてストアされる）。これらの文字（つまり、非スペース・マーク）はレンダリングされるとき、その前に置かれたベース文字に対してなんらかの方法で位置付けられることを目的とし、これらの文字自体はスペース位置を占めない。制御文字はユニコード標準でコード化されているが、制御文字自体はグラフィック文字ではない。これらの制御文字は、例えば、水平タブを示したり、フォーマッティング属性または構造などのテキストに関する追加情報を提供したり、方向性フォーマッティングによって制御したりするために使用される。ユニコード標準は双方向文字配列も用意しているので、ユニコード・コード化方式には、方向の変更を指定する文字も含まれている。例えば、ギリシア語、ローマ語およびタイ語では、左から右への方向が支配的になっているのに対し、アラビア語とヘブライ語では、右から

左への方向が支配的になっている。制御文字を使用して方向を変更することは、例えば、左から右への文字を右から左への順序で表示したいときに、ユーザまたはシステムによって要求されることがある。

従来のコード・コンバータには、1つの問題がある。それは、コンバータが1つのソース文字しか1つのターゲット文字に変換できないことである。この種の変換はある種の文字セットでは有効に働くが、多数の非スペース文字セットを含むある種の文字セット（例えば、ユニコード）の場合や、通常他の文字と関連づけられていないマークを結合するときに十分ではない。また、従来のコード・コンバータは複数の文字に関連するシンボル、合字 (ligature) または表意文字との間で変換できる機能を備えていない。

その結果として、従来のコード・コンバータにはラウンドトリップ忠実度 (round trip fidelity) がない。ラウンドトリップ忠実度とは、コード・コンバータが変換し、そのあと変換を戻してオリジナルの入力文字列を再現できる能力のことである。これが重要になるのは、コード・コンバータをある文字コードから別の文字コードに変換するときのハブとして使用するときである。

従って、複数のソース文字から単一ターゲット文字へ、あるいは単一ソース文字から複数のターゲット文字へ変換して、ラウンドトリップ文字コード変換の忠実度を保証するようなコード・コンバータが望まれている。

従来のコード・コンバータには、もう1つ問題がある。それは、ソース文字セットの文字をターゲット文字セットの文字に変換するとき方向を考慮に入れないことである。方向を考慮に入れないと、ある文字は左から右へ配列されるのに対し、他の文字は右から左へ配列されるので変換にエラーを生じることになる。これが起こる代表例として、ある与えられたソース・コード化に対して2つの等価文字をもつターゲット文字に変換し、違いが方向だけのときである。このような場合には、正しいターゲット文字をマッピングするために、ソース文字の方向が分かっている必要はない。また、従来のコード・コンバータが不十分であるのは、左から右へ配列されている（方向性）言語だけでなく、右から左へ配列されている言語にも対応する文字を含んでいるある種の文字セット（例えば、ユニコード）を取り扱えるだけの柔軟性がないためである。例えば、文字の配列ま

たは方向性がユニコード文字列内で変化するようなユニコード文字は、従来のコード・コンバータが文字列全体の方向が一定であることを前提としているので従来のコード・コンバータでは正しく変換されないことになる。

従って、文字の方向を考慮に入れながら、ソース文字セットの文字をターゲット文字セットの文字に変換することができるようなコード・コンバータが望まれている。

ある種の従来のコード・コンバータには、以下のようなとき起こる別の問題がある。ネットワークからテキストを受信するとき、そのテキストに関連するデータはデータ・ブロックで到着するのが代表的である。データが完全な形で受信されるのは、データを構成するすべてのブロックが受信されたあとだけである。受信データはバッファに置かれ、そこでデータは文字コードの変換を待っている。しかし、バッファは、多くの場合、データ・ストリーム全体を格納する能力もなければ、その1ブロックさえも格納する能力をもっていないこともある。いずれの場合も、スキャナ408によって後でテキスト要素と判断されるはずのものの途中でバッファの終わり（つまり、バッファ内の最後の文字）が現れることがある。テキスト要素の途中でバッファの終わりが現れると、スキャナ408は、後続文字が最後のテキスト要素に影響したり、その一部であることがあるので最後のテキスト要素を正しく得ることができなくなる。

大部分のコード・コンバータには、別の問題がある。それは、あるソース文字セットの文字をあるターゲット文字セットの文字に変換するときコンテキストを考慮に入れないことである。ある種の文字セット（例えば、ユニコード）では、文字セットにはコンテキスト表現形式が異なるごとに別々の文字コードが含まれているのに対し、別の時では、単一文字コードだけが用意され、コンテキストは表現形式を判断するために使用されている。しかし、従来のコード・コンバータはコードをそのコンテキストに従って変換することができない。コンテキスト・ベースのコード変換が特に問題となるのは、変換の対象となる文字セット（例えば、ユニコード）が文字コンテキストを受け入れるために、いくつかの手法を組み合わせて利用するときである。

従って、文字のコンテキストを考慮に入れながら、ソース文字セットの文字を

ターゲット文字セットの文字に正確かつ柔軟に変換できるようなコード・コンバータが望まれている。

発明の概要

概要を説明すると、本発明はコード変換および／または切り捨て処理手法に関するものである。

本発明の第1側面では、コード変換手法によれば、ラウンドトリップ忠実度が得られると共に、その結果の文字コードが他のプラットフォームと互換性をもつことが保証される。コード変換システムは単一ソース文字または文字シーケンスを単一ターゲット文字またはターゲット文字シーケンスのどちらかにマッピングする能力をもっている。ラウンドトリップ忠実度によれば、ソース・テキストはターゲット・テキストに変換し、その後オリジナル・ソース・テキストに再び戻るように変換することが可能である。互換性は標準ターゲット文字の使用を最大限にし、プライベート文字の使用を最小限にすることにより保証される。コード変換は他の文字セットからユニコード文字へ、ユニコード文字から他の文字セットへ変換するとき利用すると、特に便利である。ユニコード文字シーケンスをターゲット文字セット内の単一文字にマッピングすることは、従来は提供されていなかった。本発明は、オペレーションだけでなく、データ保管においても大幅な柔軟性が得られる強力な解決法を提供している。本発明は様々な態様で実現することができる。つまり、方法、装置またはシステムとして実現することもコンピュータ可読メディア上に実現することも可能である。

ソース文字列をターゲット文字列に変換する方法としては、本発明の第1側面による本発明の実施例は、第1文字コード化をもつソース文字列を受信し、ソース文字列をテキスト要素に順次に分割し（各テキスト要素はソース文字列の1つまたは複数の文字を含んでいる）、テキスト要素の各々ごとに第2文字コード化に関連する変換コードをマッピング・テーブルで調べ（ルックアップ）、テキスト要素の変換コードを結合して第2文字コード化のターゲット文字列を形成するオペレーションを実行する。

さらに、本発明の第1側面によれば、マッピング・テーブルにはレギュラ・

マッピングとフォールバック・マッピングを含めることが可能であり、ルックアップ・オペレーションは、レギュラ・マッピングを使用するテキスト要素の変換コードがマッピング・テーブルにないとき、フォールバック・マッピングを使用するテキスト要素の各々の変換コードを判断することが可能である。また、好ましいことは、文字の各々にそれぞれに関連する文字クラスをもたせ、少なくともその一部がソース文字列内の文字の文字クラスに基づいてソース文字列の分割が行われるようにすることである。

ソース文字列をターゲット文字列に変換するコード変換システムとしては、本発明の第1側面による本発明の実施例は、第1文字コード化をもつソース文字列を第2文字コード化をもつターゲット文字列に変換することを制御するコンバータ、ソース文字列をテキスト要素（各テキスト要素はソース文字列の1つまたは複数の文字を含んでいる）に分割するスキャナ、ソース・コード化のテキスト要素に対するターゲット・コード化をストアしておくマッピング・テーブルおよびテキスト要素の各々ごとに第2文字コード化に関連する変換コードをマッピング・テーブルで調べる（ルックアップ）ルックアップ・テーブルとを含んでいる。

本発明の第1側面によるコード変換システムには、さらに、フォールバック・ハンドラとスキャナ・テーブルを含めることが可能である。フォールバック・ハンドラは、ルックアップ・ハンドラが1つまたは複数のテキスト要素の変換コードを提供することができないような、特殊な場合にフォールバック変換コードを提供する。フォールバック変換コードはテキスト要素内の文字と正確には等価ではないが、類似したグラフィック外観をもつターゲット・コード化内の1つまたは複数のコード・ポイントを含んでいる。スキャナ・テーブルは、入力文字列内の個別文字を現在のテキスト要素内に含めるべきか、それとも新しい次のテキスト要素を開始すべきかをスキャナが判断するのを支援するものである。

入力文字列をスキャンするスキャンング・システムとしては、本発明の第1側面による本発明の実施例は、入力文字を入力文字列から取得する入力デバイス（入力文字列は文字コード化をもち、入力文字列の各文字は文字クラスをもっている）、入力文字の属性を提供する属性テーブルおよびステート・マシンの次の

状態と次のアクションの両方を、入力文字の次の状態とステート・マシンの現在状態に従って判断するステート・マシンを含んでいる。スキヤニング・システムは入力文字列の入力文字を現在のテキスト要素内に含めるべきか、現在のテキスト要素を終わらせて次のテキスト要素を開始させるかを、ステート・マシンが判断したアクションに基づいて判断する。好ましくは、属性には、入力文字の文字クラスが含まれおり、ステート・マシンは入力文字の文字クラスとステート・マシンの現在状態に基づいて次の状態と次のアクションを判断する。

ソース文字列をターゲット文字列に変換するためのプログラム命令を収めておくコンピュータ可読メディアとしては、本発明の第1側面による本発明の実施例は、第1文字コード化をもつソース文字列を受信することをコンピュータに実行させるように構成されたコンピュータ可読コード、ソース文字列をテキスト要素（各テキスト要素はソース文字列の1つまたは複数の文字を含んでいる）に分割することをコンピュータに実行させるように構成されたコンピュータ可読コード、テキスト要素の各々ごとに第2文字コード化に関連する変換コードを調べる（ルックアップ）ことをコンピュータに実行させるように構成されたコンピュータ可読コード、およびテキスト要素の変換コードを結合して第2文字コード化のターゲット文字列を形成することをコンピュータに実行させるように構成されたコンピュータ可読コードを含んでいる。

本発明の第2側面では、コード変換システムはソース文字コード化からの文字をターゲット文字コード化に変換するとき方向を考慮に入れる。

本発明の第2側面によるコード変換システムは単一ソース文字または文字シーケンスを単一ターゲット文字またはターゲット文字シーケンスのどちらかにマッピングする能力を備えている。変換される文字の方向を判断または解明することにより、コード変換システムは判断または解明された文字の方向を利用して、ターゲット文字・コード化への正しいマッピングが利用されることを保証する。従って、本発明によれば、ソース文字列内の文字の方向性が変化したときでも正しいコード変換が達成される。本発明は様々な態様で実現することが可能である。方法、装置またはシステムとして実現することも、コンピュータ可読メディア上に実現することも可能である。

ソース文字列をターゲット文字列に変換するコード変換システムとしては、本発明の第2側面による本発明の実施例は、ソース文字コード化をもつ入力文字列をターゲット文字コード化をもつターゲット文字列（入力文字列は複数の文字を含んでいる）に変換することを制御するコンバータ、入力文字列内の文字を判断するスキャナ、ソース・コード化の文字に対するターゲット・コード化をストアしておくマッピング・テーブル、および入力文字列内の文字の各々ごとにターゲット文字コード化に関連する変換コードを、入力文字列内の文字の方向とソース・コード化に基づいてマッピング・テーブルで調べるルックアップ・ハンドラを含んでいる。好ましくは、スキャナはさらに入力文字列をテキスト要素（各テキスト要素は入力文字列の1つまたは複数の文字を含んでいる）に分割し、スキャナはテキスト要素の方向を判断し、マッピング・テーブルはソース・コード化に対するターゲット・コード化をストアしており、ルックアップ・ハンドラはテキスト要素の各々ごとにターゲット文字コード化を、テキスト要素内の文字の方向とソース・コード化に基づいて調べる。この実施例には、前記ルックアップ・ハンドラが1つまたは複数の変換コードを提供できないような、特定な場合にはフォールバック変換コードを提供するフォールバック・ハンドラを含めることも可能である。

ソース文字列をターゲット文字列に変換する方法としては、本発明の第2側面による本発明の実施例は、第1文字コード化をもつソース文字列を受信し（ソース文字列は複数のソース文字を含んでいる）、ソース文字列のソース文字の方向を判断し、ソース文字の各々ごとに第2文字コード化に関連する変換コードを、第1文字コード化と判断された方向に基づいてマッピング・テーブルで調べ、ソース文字の変換コードを結合して第2文字コード化のターゲット文字列を形成するオペレーションを実行する。

ソース文字列をターゲット文字列に変換するためのプログラム命令を収めておくコンピュータ可読メディアとしては、本発明の第2側面による本発明の実施例は、第1文字コード化をもつソース文字列を受信することをコンピュータに実行させるように構成されたコンピュータ可読コード、ソース文字列内のソース文字の各々の方向を判断し、ソース文字列をテキスト要素（各テキスト要素はソース

文字列の1つまたは複数の文字を含んでいる)に分割することをコンピュータに実行させるように構成されたコンピュータ可読コード、テキスト要素の各々ごとに第2文字コード化に関連する変換コードを調べることをコンピュータに実行させるように構成されたコンピュータ可読コード、およびテキスト要素の変換コードを結合して第2文字コード化のターゲット文字列を形成することをコンピュータに実行させるように構成されたコンピュータ可読コードを含んでいる。

本発明の第3側面では、コード変換システムはソース文字コード化からの文字をターゲット文字コード化に変換するときコンテキストを考慮に入れる。

本発明の第3側面によるコード変換システムは単一ターゲット文字またはターゲット文字シーケンスを単一ターゲット文字またはターゲット文字シーケンスのどちらかに変換する能力を備えている。変換される文字のコンテキストを判断することにより、コード変換システムは判断された文字のコンテキストを利用してターゲット文字コード化への正しいマッピングが利用されることを保証する。従って、本発明によれば、文字のコンテキストが異なるターゲット・コード化に導くときでも正しいコード変換が達成される。本発明は様々な態様で実現することが可能である。方法、装置またはシステムとして実現することも、コンピュータ可読メディア上に実現することも可能である。

ソース文字列をターゲット文字列に変換するコード変換システムとしては、本発明の第3側面による本発明の実施例は、ソース文字コード化をもつ入力文字列(入力文字列は複数の文字を含んでいる)をターゲット文字コード化をもつターゲット文字列に変換することを制御するコンバータ、入力文字列内の文字の各々のコンテキストを判断するスキャナ、ソース・コード化の文字に対するターゲット・コード化をストアしておくマッピング・テーブル、および入力文字列内の文字の各々ごとにターゲット文字コード化に関連する変換コードを、コンテキストと入力文字列内の文字のソース・コード化に基づいて前記マッピング・テーブルで調べるルックアップ・ハンドラを含んでいる。好ましくは、スキャナはさらに入力文字列をテキスト要素(各テキスト要素は入力文字列の1つまたは複数の文字を含んでいる)に分割する。この実施例には、ルックアップ・ハンドラが1つまたは複数のテキスト要素の変換コードを提供できないような、特定の場合には

フォールバック変換コードを提供するフォールバック・ハンドラを含めることも可能である。

ソース文字列をターゲット文字列に変換するコード変換システムとしては、本発明の第3側面による本発明の実施例は、ソース文字コード化をもつソース文字列をターゲット文字コード化をもつターゲット文字列に変換することを制御するコンバータ手段、ソース内の文字の各々のコンテキストを判断するステート・マシン、ソース文字コード化の文字に対するターゲット文字コード化をストアしておくマッピング手段、およびソース文字列内の文字の各々ごとにターゲット文字コード化に関連する変換コードを前記マッピング手段で調べるルックアップ・ハンドラ手段を含んでいる。

ソース文字列をターゲット文字列に変換する方法としては、本発明の第3側面による本発明の実施例は、第1文字コード化をもつソース文字列を受信し（ソース文字列は複数のソース文字を含んでいる）、ソース文字列のソース文字の各々のコンテキストを判断し、ソース文字の各々ごとに第2文字コード化に関連する変換コードを、第1文字コード化とソース文字について判断されたコンテキストに基づいてマッピング・テーブルで調べ、ソース文字の変換コードを結合して第2文字コード化のターゲット文字列を形成するオペレーションを実行する。

ソース文字列をターゲット文字列に変換するためのプログラム命令を収めておくコンピュータ可読メディアとしては、本発明の第3側面による本発明の実施例は、第1文字コード化をもつソース文字列を受信することをコンピュータに実行させるように構成されたコンピュータ可読コード、ソース文字列内のソース文字の各々のコンテキストを判断し、ソース文字列をテキスト要素（各テキスト要素はソース文字列の1つまたは複数の文字を含んでいる）に分割することをコンピュータに実行させるように構成されたコンピュータ可読コード、テキスト要素の各々ごとに第2文字コード化に関連する変換コードを調べることをコンピュータに実行させるように構成されたコンピュータ可読コードおよびテキスト要素の変換コードを結合して第2文字コード化のターゲット文字列を形成することをコンピュータに実行させるように構成されたコンピュータ可読コードを含んでいる。

本発明の第4側面では、切り捨て処理手法は、変換のために受信されたソース文字列が、そのソース文字列が変換のためにソース文字列を格納している入力バッファの長さを越えたときでも、異なるターゲット文字コード化に正確に変換されることを保証する。

本発明の第4側面による切り捨て処理は、入力バッファに置かれているソース文字列の一部を切り捨て、その切り捨てられた部分がソース文字列内の後続文字に影響されることなくターゲット文字コード化に変換されるように作用する。当然のことながら、入力バッファに置かれていて、切り捨てが行われた後のソース文字列の部分（切り捨てられた部分）は最大限にしておく、コード変換が効率よく行えるようになる。本発明は、入力ソース文字列がネットワーク経由で受信されるデータであるときに特に便利である。例えば、データはインターネット経由で電子的に転送されるテキスト・データにすることができる。いずれの場合も、本発明は種々の態様で実現することが可能である。方法、装置またはシステムとして実現することも、コンピュータ可読メディア上に実現することも可能である。

ソース文字列をターゲット文字列に変換するコード変換システムとしては、本発明の第4側面による本発明の実施例は、第1文字コード化をもつソース文字列を第2文字コード化をもつターゲット文字列に変換することを制御するコンバータ、ソース文字列の一部を一度に受け入れるバッファ（ソース文字列は1つまたは複数の文字を含んでいる）、ソース文字列の一部を切り捨てるトランケータ、ソース文字列の切り捨てられた部分をテキスト要素（各テキスト要素はソース文字列の切り捨てられた部分の1つまたは複数の文字を含んでいる）に分割するスキャナ、ソース・コード化のテキスト要素に対するターゲット・コード化をストアしておくマッピング・テーブル、およびテキスト要素の各々ごとに第2文字コード化に関連する変換コードをマッピング・テーブルで調べるルックアップ・ハンドラを含んでいる。

ソース文字列を切り捨ててターゲット文字列に文字変換するための第1方法としては、本発明の第4側面による本発明の実施例は、ソース文字列のバッファ部分をバッファに受け入れ（ソース文字列は2つ以上のバッファ部分を含んでい

る)、ソース文字列の後続バッファ部分に影響されことなくターゲット文字列に変換することができる、ソース文字列のバッファ部分のサブパートを判断し、ソース文字列のバッファ部分のサブパートをターゲット・コード化に変換するオペレーションを実行する。好ましくは、第1方法は、変換対象となるバッファ部分の残余部分がサブパートと一緒にになったときバッファ部分に等しい場合には、その残余部分を次のバッファ部分またはそのサブパートと一緒に格納するオペレーションも実行する。

ソース文字を切り捨ててターゲット文字列に文字変換するための第2方法としては、本発明の第4側面による本発明の実施例は、ソース文字列の一部をバッファに受け入れ、ソース文字列のその部分内のテキスト要素(各テキスト要素はソース文字列の1つまたは複数の文字を含んでいる)を判断し、テキスト要素が完成しているかどうかを判断し、テキスト要素が完成しているときソース文字列の切り捨てられた部分にテキスト要素を挿入し、ソース文字列の部分が完全に考慮されるまでテキスト要素の判断を繰り返し、そのあと文字変換のためにソース文字列の切り捨てられた部分を出力するオペレーションを実行する。好ましくは、この第2方法は、バッファに受け入れられたソース文字列の次の部分と一緒に使用するために、ソース文字列の残余部分を格納しておくオペレーションも実行する。

ソース文字列をターゲット文字列に変換するためのプログラム命令を格納しておくコンピュータ可読メディアとしては、本発明の第4側面による本発明の実施例は、第1文字コード化をもつソース文字列の一部をバッファに受け入れることをコンピュータに実行させるように構成されたコンピュータ可読コード、ソース文字列の一部を切り捨てることをコンピュータに実行させるように構成されたコンピュータ可読コード、ソース文字列の切り捨てられた部分をテキスト要素(各テキスト要素はソース文字列の切り捨てられた部分の1つまたは複数の文字を含んでいる)に分割することをコンピュータに実行させるように構成されたコンピュータ可読コード、テキスト要素の各々ごとに第2文字コード化に関連する変換コードを調べることをコンピュータに実行させるように構成されたコンピュータ可読コード、およびテキスト要素の変換コードを結合して第2文字コード化の

ターゲット文字列を形成することをコンピュータに実行させるように構成されたコンピュータ可読コードを含んでいる。

本発明のその他の側面および利点は、本発明の原理を例示している添付図面を参照して以下に詳述している説明の中で明らかにする。

図面の簡単な説明

本発明の理解を容易にするために、以下では、添付図面を参照して本発明を詳しく説明する。なお、図面において、類似する構成要素は類似の参照符号を付けて示されている。

図1は、本発明による代表的なコンピュータ・システムを示すブロック図である。

図2は、ユニコード文字コード化のフォーマットを示す図である。

図3は、ソース文字列を受信し、ターゲット文字列を出力する本発明による基本的ユニコード・コード変換システムを示すブロック図である。

図4は、本発明の実施例によるユニコードからのコード変換システムの実施例を示すブロック図である。

図5は、ユニコード・コード変換システムのマッピング・テーブルの好ましい配列を示す概略図である。

図6Aは、本発明の実施例によるユニコード・コード変換システムを利用するアプリケーション・プログラムによって実行される処理を示すフローチャートである。

図6Bは、本発明の実施例による切り捨て処理を示すフローチャートである。

図7は、本発明の実施例によるユニコード・コンバータ処理を示すフローチャートである。

図8は、本発明の実施例による更新オフセット処理を示すフローチャートである。

図9Aおよび図9Bは、本発明の実施例による次のテキスト要素処理を示すフローチャートである。

図10A、図10Bおよび図10Cは、本発明の実施例による次のテキスト要

素処理を示すフローチャートである。

図11は、本発明の実施例によるスキナを示すブロック図である。

図12は、図10に示す属性テーブルの好ましいフォーマットを示す概略図である。

図13は、本発明の実施例による属性ルックアップ処理を示すフローチャートである。

図14Aおよび図14Bは、本発明の好適実施例による次のアクションを判断するためにスキナによって利用されるスキナ・テーブルを示す概略図である。

図15は、本発明の実施例による好ましいレイアウトと、スキナ・テーブルにストアされる情報の両方を表しているテーブルである。

図16Aおよび図16Bは、本発明の好適実施例による好ましいレイアウトと、スキナ・テーブルにストアされる情報の両方を表しているテーブルである。

図17Aは、本発明の実施例による方向解明処理を示すフローチャートである。

図17Bないし図17Dは、本発明による双方向状態テーブルの好ましいレイアウトを表しているテーブルである。

図18は、本発明の実施例による初期方向処理を判断するためのフローチャートである。

図19は、本発明の実施例によるルックアップ・テキスト要素処理を示すフローチャートである。

図20は、本発明の実施例によるチェーン・フォーマット処理を示すフローチャートである。

図21は、本発明の実施例による範囲フォーマット処理を示すフローチャートである。

図22は、本発明の実施例によるリスト・フォーマット処理を示すフローチャートである。

図23Aおよび図23Bは、本発明の実施例によるセグメント配列フォーマット

ト処理を示す図である。

図24は、本発明の実施例による変形リスト・サーチ処理を示すフローチャートである。

図25Aおよび図25Bは、本発明の実施例による変形リスト処理に関連するテーブルとデータ・フォーマットを示す概略図である。

図26A、図26Bおよび図26Cは、本発明の実施例による出力シーケンスへのルックアップ・ターゲットのマッピング処理を示すフローチャートである。

図27は、本発明の実施例による本発明の処理に従うフォールバック・ハンドリング処理を示すフローチャートである。

図28は、本発明の実施例によるデフォルト処理を示すフローチャートである。

図29は、本発明の実施例によるユニコードへのコード変換の実施例を示すブロック図である。

発明の最良の実施形態

以下、図1ないし図29を参照して本発明の実施例について説明する。なお、この分野の精通者ならば理解されるように、これらの図を参照して以下に詳述する説明は本発明の理解を容易にすることを目的とし、本発明は以下に説明する実施例に限定されるものではない。

本発明の第1側面によるコード変換システムはソース文字を異なるコード化のターゲット文字に変換するものである。本発明によるコード変換システムはラウンドトリップ忠実度を備えていると共に、その結果の文字コードが他のプラットフォームと互換性をもつことを保証する。コード変換システムは単一ソース文字または文字シーケンスを単一ターゲット文字またはターゲット文字シーケンスのどちらかにマッピングする機能を備えている。ラウンドトリップ忠実度により、ソース・テキストはターゲット・テキストに変換し、そのあとオリジナル・ソース・テキストに戻すように再変換することができる。その結果の文字コードが他のプラットフォームと互換性をもつことは、標準ターゲット文字の使用を最大限にし、プライベート文字の使用を最小限にすることによって保証される。コード

変換システムは他の文字セットからユニコード文字に変換し、ユニコード文字を他の文字セットに変換するとき利用すると、特に便利である。ユニコード文字シーケンスをターゲット文字セット内の単一文字にマッピングすることは従来にはなかったものである。

コード変換システムはコンピュータ・システムや他のエレクトロニック・デバイスにしてこれらのコード変換オペレーションを行うことができる。このコンピュータ・システムは必要とする目的だけに使用するように構築することも、コンピュータ・プログラムに従って動作する汎用コンピュータにすることも可能である。以下に説明する処理はどのコンピュータ・システムや他のエレクトロニック・デバイスにも適用可能である。具体的には、種々の汎用コンピューティング・マシンは以下に教示する説明に従って書かれたソフトウェアと共に使用することができるが、もっと特殊化されたエレクトロニック・デバイスを構築すると、必要とするオペレーションを実行できるので好都合である。

本発明の第2側面によるコード変換システムはソース文字コード化からの文字をターゲット文字コード化に変換するとき方向を考慮に入れる。このコード変換システムは単一ソース文字または文字シーケンスを単一ターゲット文字またはターゲット文字シーケンスにマッピングする機能を備えている。変換される文字の方向を判断または解明することにより、コード変換システムは判断または解明された文字の方向を利用してターゲット文字コード化への正しいマッピングが得られることを保証する。従って、本発明によれば、ソース文字列内の文字の方向が変化するときでも正しいコード変換が達成される。

本発明がアラビアまたはヘブライ・ベースの文字セットが使用されるときに特に便利であるのは、これらの文字が右から左への方向をもっているからである。本発明によれば、どちらかの方向を取り扱うときの柔軟性が得られるだけでなく方向を途中で変更できるという機能も得られる。方向を途中で変更できることはアラビア文字および／またはヘブライ文字が左から右への方向が最も普通である他の文字セットと一緒に使用されるような場合に重要である。異なる方向をもつ文字の使用例として、スペース文字がある。ユニコードでは、コード化は1つだ

けであり、固有の方向をもっていない。他方、M a c A r a b i cでは、左から右へのスペース文字と右から左へのスペース文字の2種類がある。

変換されるソース文字の方向を判断または解明することは、方向解明手法によって達成されるが、この手法についてはソース文字を異なるコード化のターゲット文字に変換するコード変換システムと関連づけて以下で詳しく説明する。

本発明の第3側面によるコード変換システムはソース文字コード化からの文字をターゲット文字コード化に変換するときコンテキストを考慮に入れる。このコード変換システムは単一ソース文字または文字シーケンスを単一ターゲット文字またはターゲット文字シーケンスのどちらかにマッピングする機能を備えている。文字のコンテキストを判断することにより、コード変換システムは判断された文字のコンテキストを利用してターゲット・コード化への正しいマッピングが得られることを保証する。従って、本発明によれば、文字のコンテキストがターゲット・コード化へのマッピングに影響するときでも正しいコード変換が達成される。

ある種の文字セットでは、文字が使用されているコンテキストによって文字が異なる表現形態をもつことになる。これらの文字の表現形態は絵文字(glyphs)である。異なる表現形態は表示されるとき異なって現れる。文字が使用されているコンテキストによって表現形態が決まる。アラビア文字はそのコンテキストに応じて異なる表現形態をもつ言語である。

アラビア・スクリプトはアラビア語を書くために使用される。アラビア・スクリプトはその印刷形体においても筆写体(カーシブ)である。その結果として、同じ文字はそれが隣の文字とどのように結合されるかに応じて異なった形体で書かれることがある。このような文字の例として、アラビア文字“H E H”(ユニコード文字ではu 0 6 4 7とコード化される)がある。例えば、ユニコードでのアラビア文字はD O Sアラビア文字の4つの異なる表現形態の1つをコンテキストに応じてマッピングされる場合がある。その結果、本発明によるコード変換システムはユニコード・ストリーム内のアラビア文字のコンテキストを判断し、正しいマッピングが得られるように動作する。

変換されるソース文字のコンテキストの判断はコンテキスト処理によって達成されるが、このコンテキスト処理はソース文字を異なるコード化のターゲット文字に変換するコード変換システムと関連づけて以下で詳しく説明する。

本発明の第2および第3側面によれば、コード変換システムはソース文字を異なるコード化のターゲット文字に変換する。本発明によるコード変換システムはラウンドトリップ忠実度を提供すると共に、その結果の文字コードが他のプラットフォームと互換性をもつことを保証する。コード変換システムは単一ソース文字または文字シーケンスを単一ターゲット文字またはターゲット文字シーケンスのどちらかにマッピングする機能を備えている。ラウンドトリップ忠実度によると、ソース・テキストはターゲット・テキストに変換し、そのあとオリジナル・ソース・テキストに戻るよう再変換することができる。その結果の文字コードと他のプラットフォームとの互換性は標準ターゲット文字の使用を最大限にし、プライベート文字の使用を最小限にすることによって保証される。コード変換システムは他の文字セットからユニコード文字に変換し、ユニコード文字を他の文字セットに変換するときに利用すると、特に便利である。ユニコード文字シーケ

ンスをターゲット文字セット内の単一文字にマッピングすることは従来にはなかったものである。

コード変換システムはコンピュータ・システムや他のエレクトロニック・デバイスにしてこれらのコード変換オペレーションを行うことができる。このコンピュータ・システムは必要とする目的だけに使用するように構築することも、コンピュータ・プログラムに従って動作する汎用コンピュータにすることも可能である。以下に説明する処理はどのコンピュータ・システムや他のエレクトロニック・デバイスにも適用可能である。具体的には、種々の汎用コンピューティング・マシンは以下に教示する説明に従って書かれたソフトウェアと共に使用することができるが、もっと特殊化されたエレクトロニック・デバイスを構築すると、必要とするオペレーションを実行できるので好都合である。

本発明の第4側面では、切り捨て処理手法は変換のために受信されたソース文字列が、そのソース文字列が変換のためにソース文字列を格納している入力バッ

ファの長さを越えたときでも、異なるターゲット文字コード化に正確に変換されることを保証する。切り捨て処理手法は入力バッファに置かれているソース文字列の一部を切り捨てて、切り捨てられた部分がソース文字列内の後続文字に影響されることなくターゲット文字コード化に変換されるように作用する。当然のことであるが、入力バッファに置かれていて、切り捨てられた後のソース文字列の部分（切り捨てられた部分）を最大限にすると、コード変換を効率よく実行することができる。

本発明による切り捨て処理手法はソース文字を異なるコード化のターゲット文字に変換するコード変換システムと関連づけて以下で詳しく説明する。

図1は本発明による代表的なコンピュータ・システム100を示すブロック図である。コンピュータ・システム100は中央処理ユニット（CPU）102を装備し、このCPUは双方向にランダムアクセス・メモリ（RAM）102と、単方向にリードオンリ・メモリ（ROM）106と結合されている。代表例として、RAM104はプログラミング命令とデータを格納し、この中には現在CPU102上で実行中のプロセスに対する他のデータや命令のほかに、以下で説明するテーブルが含まれている。代表例として、ROM106はコンピュ-

タ・システムがその機能を実行するために使用する基本的操作命令、データおよびオブジェクトを格納している。そのほかに、ハードディスク、CD ROM、磁気光学（フロッピカル）ドライブ、テープ・ドライブなどの大量記憶装置108は双方向にCPU102と結合されている。大量記憶装置108は一般的にCPUによって活発に使用されることがない追加のプログラミング命令、データおよびテキスト・オブジェクトを収容しているが、アドレス空間は例えば仮想メモリなどのためにCPUによってアクセス可能である。上述したコンピュータの各々はさらに入出力ソース110を含んでおり、その代表例として、キーボードやポインタ・デバイス（例えば、マウスやスタイラス）などの入力メディアがある。コンピュータ・システム100はデータと命令を転送できるネットワーク接続112を含むこともできる。追加の大量記憶装置（図示せず）をネットワーク接続112を経由してCPU102に接続することも可能である。コンピュータ・

システム100はさらに、コンピュータ・システム100によって生成または表示されたテキストおよびイメージ（画像など）を見るための表示スクリーン114を含んでいる。

CPU102はオペレーティング・システム（図示せず）と一緒にあって、コンピュータ・コードを実行するように動作する。コンピュータ・コードはRAM104、ROM106、または大量記憶装置108に置いておくことができる。また、コンピュータ・コードはポータブル・プログラム・メディア116に置いておき、必要時にコンピュータ・システム100にロードまたはインストールすることも可能である。ポータブル・プログラム・メディア116の例としては、CD-ROM、PCカード・デバイス、RAMデバイス、フロッピディスク、磁気テープなどがある。

I. 定義

1. コード・ポイント：コード・ポイントとは、特定のコード化におけるビット・パターンである。通常、ビット・パターンは1または2バイト以上の長さになっている。ユニコードのコード・ポイントは常に16ビットまたは2バイトである。

2. コード化：コード化とは、文字セットとコード・ポイントのセットとを1対1で対応づけたもの（マッピング）である。例えば、ASCIIコード化はa-z、A-Z、および0-9を含むセットをコード・ポイントx00-x7Fに対応づけている。

3. テキスト要素：テキスト要素は特定のオペレーションで1つの単位として扱われる1つまたは2つ以上のコード・ポイントのシーケンスである。例えば、LATIN CAPITAL LETTER Uとその後に続くNON-SPACING DIAERESISは本発明によるコード変換オペレーションの対象となるテキスト要素である（例えば、この例では、2つの隣り合う文字）。

4. 絵文字：文字を可視的に表現した表示形態。例えば、イタリックの“a”とローマン字の“a”は同じ基礎文字を表現する2つの異なる絵文字である。

5. 表現形態：表現形態はコンテキストに応じてその可視形体を変化させる絵

文字である。ある種のコード化はコンテキストから独立している抽象文字だけをマッピングするのに対し、他のコード化は表現形体だけをマッピングする。例えば、“fi”のような合字は、LATIN CAPITAL LETTER Fとその後に続くLATIN CAPITAL LETTER Iの文字シーケンスの表現形態である。

6. フォールバック：フォールバックはソース・コードに正確に等価ではないが、オリジナルの情報の一部を残しているターゲット・コード化内の1つまたは2つ以上のコード・ポイントのシーケンスである。例えば、(C)は©を表す可能なフォールバックである。

7. デフォルト：デフォルトはターゲット・コード化内にソース・コード・ポイントに類似したものがないとき使用されるターゲット・コード化内の1つまたは2つ以上のコード・ポイントのシーケンスである。

II. ユニコード・コンバータ

本発明による一般的変換手法はソース文字を異なるコード化のターゲット文字に変換する。好ましくは、ソース文字またはターゲット文字のどちらかはユニ

コード文字になっている。

ユニコード標準は開発されたいくつかの文字コード化を汎用国際的文字コード化標準に統一化したものである。図2はユニコード文字コード化のフォーマットを示す図である。具体的には、ユニコード標準に用意されているコードは、図2に示すフォーマット200で示すように16ビット幅になっている。本明細書では、ユニコード文字は頭にuを付けた16進数で表され（例えば、u001）、他のコード化内の文字は頭にxを付けた16進数で表されている（例えば、x41は1バイト文字を表し、x8140は2バイト文字を表している）。

図3は、ソース文字列302を受信し、ターゲット文字列304を出力する本発明による基本的ユニコード・コード変換システム300を示すブロック図である。ユニコード・コード変換システム300はソース文字列302の文字を、ターゲット・ストリーム内にあって、文字コード化がソース文字列で使用されたコード化と異なっている1つまたは複数の文字に変換するように動作する。好まし

くは、ユニコード・コード変換システム300はユニコードから異なるターゲット・コード化に変換するか（ユニコードから）、あるいは異なるソース・コードから変換する（ユニコードへ）。

図4は本発明によるユニコード・コード変換システム400の実施例を示すブロック図である。本発明の第4側面では、トランケータ407とバッファ405が存在し、適用されているが、第1、第2または第3側面では、これらは必ずしも存在する必要がないので、必ずしも適用されとは限らない。

本発明の第1、第2および第3側面によれば、ユニコード・コード変換システム400はユニコード文字列404を受信し、ターゲット文字列406を出力するユニコードからのコンバータ402を含んでいる。ユニコードからのコンバータ402は本発明に従ってコード変換プロセスを実行する。その際に、ユニコードからのコンバータ402はスキャナ408とやりとりする。スキャナ408はスキャナ・テーブル408と一緒に使用されて、ユニコード文字列をスキャンしテキスト要素を特定する。そのあと、ユニコードからのコンバータ402はルックアップ・ハンドラ412を使用して、スキャナ408によって特定されたテキスト要素のターゲット・コード化内の1つまたは複数の文字を調べる。ルック

アップ・ハンドラ412はマッピング・テーブル14を使用してテキスト要素のターゲット・コード化内の1つまたは複数の文字を取得する。さらに、ユニコードからのコンバータ402はフォールバック・ハンドラ416を使用することも可能である。フォールバック・ハンドラ416はマッピング・テーブル414と一緒に使用され、ルックアップ・ハンドラ412がテキスト要素のターゲット・コード化内の1つまたは複数の文字を特定できなかった場合に、ターゲット・コード化内にあって、テキスト要素のフォールバック・マッピングとして使用できる1つまたは複数の文字を特定するように動作する。状態管理機構(state administrator)418は変換の現在状態に関する情報を維持し、あるいはストアしている。この情報の例としては、シメトリック・スワッピングのコンテキスト、方向および状態がある。

本発明の第4側面によれば、ユニコード・コード変換システム400はユニコ

ード文字列404を受信し、ターゲット文字列406を出力するユニコードからのコンバータ402を含んでいる。ユニコード文字列はバッファ405にストアされる。トランケータ407はバッファ405にストアされたユニコード文字列404の長さを切り捨てて、正確な変換が行われることを保証する。

ユニコードからのコンバータ402はコード変換の全体的オペレーションを制御する。その際に、ユニコードからのコンバータ402はスキャナ408とやりとりする。スキャナ408はスキャナ・テーブル408と一緒に使用されて、切り捨てられたユニコード文字列404（トランケータ407から与えられたもの）をスキャンしテキスト要素を特定する。そのあと、ユニコードからのコンバータ402はルックアップ・ハンドラ412を使用して、スキャナ408によって特定されたテキスト要素のターゲット・コード化内の1つまたは複数の文字を調べる。ルックアップ・ハンドラ412はマッピング・テーブル14を使用してテキスト要素のターゲット・コード化内の1つまたは複数の文字を取得する。さらに、ユニコードからのコンバータ402はフォールバック・ハンドラ416を使用することも可能である。フォールバック・ハンドラ416はマッピング・テーブル414と一緒に使用され、ルックアップ・ハンドラ412がテキスト要素のターゲット・コード化内の1つまたは複数の文字を特定できなかった

場合に、ターゲット・コード化内にあって、テキスト要素のフォールバック・マッピングとして使用できる1つまたは複数の文字を特定するように動作する。状態管理機構(state administrator)418は変換の現在状態に関する情報を維持し、あるいはストアしている。この情報の例としては、シメトリック・スワッピングのコンテキスト、方向および状態がある。この種の情報は、切り捨てられたユニコード文字列がブロックの終わりで終わっていない非ブロック区切り変換を行うとき必要になるものである。そのような場合には、変換の現在状態に関する情報をストアしておく、コード変換プロセスは入力文字列404がバッファ405のサイズよりも大きいときでも、正確なコード変換を行うことができる。

A. スキャナおよびスキャナ・テーブル

スキャナ408はスキャナ・テーブル410と一緒に使用されて、ユニコード

文字列404をスキャンし、ルックアップ・ハンドラ412が必要とする次のテキスト要素と追加情報を戻す。追加情報には、方向情報、コンテキスト情報、および種々の状態インジケータの1つまたは2つ以上が含まれている。以下では、スキャナ408の一般的オペレーションについて説明する。スキャナ408は入力ユニコード文字列404の文字をスキャンして行く。ターゲット・コード化のために方向情報が必要であれば、テキスト要素内の各文字ごとに文字方向が取得される。また、ターゲット・コード化のために文字コンテキスト情報が必要であれば、テキスト要素内の各文字ごとに文字コンテキスト情報が取得される。その後、スキャナ408が文字の各々をスキャンしていくとき、スキャナ408はスキャナ・テーブル410に入っている情報に従って文字に対するアクションをとる。スキャナ408がどのようなアクションをとるかは、状態と文字クラスに基づいて判断される。スキャナ408がとることができるアクションとしては、現在の文字にマークを付けること、シメトリック・スワッピング・ビットをセットまたはクリアすること、テキスト要素のコンテキスト形式を記録すること、テキスト要素が再配列を必要とすることを示すフラグをセットすること、テキスト要素の終わりを示すこと、などがある。シメトリック・スワッピング・ビット、

コンテキストおよび方向はスキャナの状態に関する情報として状態管理機構418によってセーブされる。戻す前に、スキャナ408はテキスト要素のコンテキスト情報をセーブしておく。スキャナ408はテキスト要素（入力文字列内の各テキスト要素）とその属性を返却する。属性には次のものがある。方向、クラス、優先順位、シメトリック・スワッピング状態、サブセットおよびコンテキストである。スキャナ408がテキスト要素を判断したあと、文字を標準形順序に再配列する必要がある場合がある。1つの例として、テキスト要素内の文字の再配列はユニコードで定義されている標準形順序になっていない非スペース・マークがテキスト要素に含まれているときに行われる。

好ましくは、スキャナ408はスキャナ・テーブル410と一緒に、並列に動作するペアのステート・マシンとして実現されている。第1ステート・マシンは文字方向を解明し、第2ステート・マシンは該当する場合には、テキスト要素と

文字形式コンテキスト情報を計算し、シメトリック・スワッピング状態も記録しておく。別々になった2つのステート・マシンを使用すると、ユニコード・コード変換システム400の設計と保守が容易化される。第1および第2ステート・マシンは状態とクラスによってインデックスされる2次元配列（またはテーブル）として実現することができる。スキャナ408がとるべきアクションが文字方向によって決まる場合には、ステート・マシン・エントリは各方向についてスキャナ408がとるべき該当アクションを収めている別のテーブルまでのインデックスである。

スキャナ408の機能は入力ユニコード文字列404をテキスト要素に変換し、テキスト要素とその属性を戻すことである。スキャナ408はテキスト要素のある種の特性をセーブしておく必要があり、そうすれば、ターゲット・コード化で正しく変換されることになる。すなわち、特性には、方向、コンテキストおよびシメトリック・スワッピング状態がある。しかし、スキャナ408は、そのオペレーションが特定のターゲット・コード化から独立しているので、どのようなターゲット・コード化であるかを知っている必要はない。それにもかかわらず、ユニコード変換システム400は、テキスト要素の定義（つまり、チャンク行動）が、スキャナ・テーブル410を変更するだけでターゲット・コード化と

共に変化できるように実現されていることが好ましい。

文字の方向性は文字を表示するために使用される。例えば、アラビアまたはヘブライ文字が表示スクリーンに表示されるとき、これらは右から左への順序になっている。大部分のユニコード文字は黙示的方向をもっている（ユニコード・バージョン1.0の407ページ（セクション4.6）および611ページ（付録A）を参照）。ユニコード標準に用意されている暗黙的方向クラスとその値には次のものがある。左から右へ（0）、右から左へ（1）、ヨーロッパ数字（2）、ヨーロッパ数字セパレータ（3）、ヨーロッパ数字ターミネータ（4）、アラビア数字（5）、共通数字セパレータ（6）、ブロック・セパレータ（7）、セグメント・セパレータ（8）、ホワイトスペース（9）およびその他の数字である。スキャナ408はテキスト要素の文字の方向クラスを調べる。次に、その方

向クラスはテキスト要素の方向を説明するために使用される。また、方向性をオーバーライドまたは埋め込ませる特殊なユニコード文字もある。これらの特殊なユニコード文字はスキャナ408によって単一文字テキスト要素として扱われる。

テキスト要素を形成するときスキャナ408が従う基本的ルールがいくつかある。ベース・ルールでは、適用されるルールがなければ、テキスト要素は単一ユニコード文字とされる。別のルールでは、ベース文字に続く非スペースまたは結合マークはそのベース文字と一緒に単一テキスト要素として分類される。さらに別のルールでは、シンボル（つまり、朝鮮ハングル・ジャモス文字）、合字または表意文字に関連する文字が見つかったと、これらはテキスト要素に結合される。さらに別のルールでは、フラクシオン・スラッシュの両側が1つまたは複数の10進数のシーケンスで囲まれていると、これらは数字フラクシオン・テキスト要素として結合される。

以下では、非スペースまたは結合マークに関するルールについて詳しく説明する。ユニコード標準によれば、非スペース・マークはベース文字のあとに置かれている。従って、ベース文字のあとに置かれた非スペース文字はベース文字を含むテキスト要素の一部となる（ユニコード標準バージョン1.0、403ページ（セクション4.3）を参照）。例えば、単一の非スペース文字の後に非スパー

ス文字でない文字が続いているときは、その非スペース文字は直前の文字と一緒にテキスト要素として結合される。その場合、テキスト要素の長さは2であり、テキスト要素の属性はベース文字によって定義されている。前に置かれた文字がなければ、非スペース文字は単一テキスト要素として渡されるだけである。複数の非スペース文字もこのようにして結合することができる。

以下では、朝鮮ハングル・ジャモス文字について詳しく説明する。各ハングル文字は暗黙値を持ち、これは次のクラスの1つになっている。Choseong（初期）、Jungseong（中間）またはJongseong（最終）である。ユニコード標準バージョン1.1（セクション5）はこれらの文字のコードと許容される組み合わせをリストしている。スキャナ408はこれらの文字の許容される組み合わせに従って朝鮮ジャモス文字をグループ化する。入力が許容さ

れない組み合わせのときは、スキャナ408は文字を単一テキスト要素として返却する。前述したように、ハングル分節のあとに結合マークが付いているときは、その結合マークはハングル分節のテキスト要素内に挿入される。

次に、数字フラクシオンに関するルールについて詳しく説明する。スキャナ408は最初にフラクシオン・スラッシュ数字の各文字を、それらが単一文字テキスト要素であるものとして取り扱う。しかし、完全なフラクシオン・スラッシュ・シーケンスが見つかったときは、スキャナ408はそのシーケンスに関連する文字を単一テキスト要素になるように結合する。ディジットが結合マークと一緒に見つかったときは、そのディジットと結合マークはフラクシオン・スラッシュの一部となることができないが、ディジットと結合マークと一緒にテキスト要素を形成することができる。

非スペース文字を除き、すべてのアラビア文字は単一テキスト要素としてスキャナ408を通過する。アラビア形式のシェーピング状態文字も、単一テキスト要素としてスキャナを通過する。方向フォーマッティング・コードは単一テキスト要素としてスキャナ408を通過する。

B. ルックアップ・ハンドラ、マッピング・テーブルおよびフォールバック・ハンドラ

マッピング・テーブル414は1つまたは複数のユニコード文字の入力シーケンスをターゲット・コード化における1つまたは複数の出力シーケンスと突き合わせるためにルックアップ・ハンドラ412によって使用される。ユニコード・シーケンス（つまり、テキスト要素）自体のほかに、入力シーケンスに関するある種の追加情報が得られ（例えば、方向、今テキスト、シメトリック・スワッピング状態、垂直形式要求、フォールバック要求、許容範囲、変種）、ある種のテーブルはこの情報を利用している。好ましくは、マッピング・テーブル414はフォールバック・ハンドラ416が必要とするデータもストアしているが、別のテーブルを用意してフォールバック・ハンドラ416に使用させることも可能である。

図5はユニコード・コード変換システム400のマッピング・テーブル414

の好ましい配列を示す概略図である。マッピング・テーブル414は好ましくはヘッダ部分500を含み、マッピング・テーブル414のデータのセグメントはテキスト要素内の文字数に基づいて分割されている。ヘッダ部分500の内容は以下で詳しく説明する。図5に示すマッピング・テーブル414は1からN文字までのテキスト要素のコード化をサポートしている。ルックアップ・ハンドラ412が1文字テキスト要素のターゲット・コード化を探すためにマッピング・テーブル414をサーチするとき、マッピング・テーブル414のセグメント502が使用される。同様に、テキスト要素が2文字幅であれば、セグメント504が使用され、テキスト要素がN文字幅であれば、セグメント506が使用される。図4と図5は単一マッピング・テーブル414を示しているが、ユニコード・コード変換システム400は複数の異なるマッピング・テーブル414を使用する。つまり、各ターゲット文字セットごとに1つのマッピング・テーブルを使用する。各マッピング・テーブルは複数のサブテーブルを含んでいる。

マッピング・テーブル414はサイズと全体的変換速度要件を考慮に入れて設計されている。マッピング・テーブル414はルックアップ時間を重大に低下させることなく可能な限り小さくしておくべきであり、ルックアップ時間はテー

ブル・サイズを大幅に大きくすることなく可能な限り高速にしておくべきである。ユニコード・コード変換システム400は複数のテーブル・フォーマットをサポートしているので、各サブテーブルごとに異なるフォーマットにすることが可能である。そのようにすると、速度とサイズのトレードオフを特定のテーブルに合わせて必要時に調整することができる。好ましくは、マッピング・テーブル414の設計は単一ユニコード文字からターゲット・コード化内の単一文字にマッピングするのを可能な限り高速化するものでなければならないが、これは最も普通の使い方であるからである。

マッピング・テーブル414の設計は、テーブルがフォールバック・ハンドラ416の要求の少なくとも一部をサポートし、複数のマッピング許容範囲をサポートし、複数のターゲット文字セットの変種をサポートするようになっている。テーブル・フォーマットは1つまたは複数のユニコード文字シーケンスをゼロま

たは1個以上の文字の出力シーケンスにマッピングすることもできる。マッピング・テーブル414は単一入力シーケンスに対して起こり得る複数の出力シーケンスを指定することが可能であり、特定の出力シーケンスは方向、コンテキストおよびシメトリック・スワッピング状態などの属性によって判断される。本発明の第3側面に関しては、中心となるのは出力シーケンスの1つをコンテキストに基づいて選択することである。マッピング・テーブルは容易に拡張することができるので、ユニコード・コード変換システム400のコード化の振舞を容易にカスタマイズすることができる。

マッピング・テーブル414が必要とする情報には次のものがある。スキャナ408からのテキスト要素（つまり、結合マークを標準形順序にして変換される入力文字シーケンス）、垂直形式が使用可能なときに水平形式の代わりに使用するかどうか、解明されたテキスト要素の方向、テキスト要素の文字形式コンテキスト情報（初期、中間、最終、または隔離）、シメトリック・スワッピングの現在状態、どのレベルのルックアップを起動させるかの情報（許容レベル（厳格または緩和）とフォールバック（オンまたはオフ））、および特定のコード化変種の識別子である。ルックアップ・レベルに関する情報はコールまたはユニコード・コード変換システム400をコールするアプリケーション・プログラムから

与えられる。方向およびコンテキストが重要でない言語または文字では、解明された方向とコンテキスト情報は必要でない。

変種の定義、ユニコード・シーケンスからターゲット・シーケンスへの実際のマッピング、およびこれらにアクセスするために使用されるテーブル・フォーマットはマッピング・テーブル414の設計によって変更可能である。従って、正確性および若干であるが、パフォーマンスとサイズとのトレードオフはマッピング・テーブル414の設計に大きく依存している。好ましいことは、マッピング・テーブル414が厳格および緩和マッピング、フォールバック・マッピング、およびデフォルト・マッピングをサポートすることである。

厳格マッピングはラウンドトリップ忠実度が保証されるコード変換である。ユニコードからターゲット文字セットへの厳格マッピングはその文字セットからユ

ニコードへのユニコードへのマッピングとは逆である。ユニコードからターゲット文字セットへの緩和マッピングはターゲット文字セット内の文字の定義または隔離された用途の範囲に属する追加マッピングである。緩和マッピングは正しくマッピングされるように見えるが、若干のあいまいさがある。例えば、多くの文字セットでは、単一文字は、明示的定義、あいまいな定義、または確立された用法のいずれかによって複数の意味を持つことがある。例えば、Shift-JIS文字x8161は2つの意味を持つように規定されている。すなわち、「2重垂直線」と「平行」である。これらの意味の各々は異なるユニコード文字u2016「2重垂直線」とu2225「に平行」に対応している。Shift-JISからユニコードにマッピングするときは、コード変換システムはこれらのユニコード文字の1つ、つまり、2重垂直線を選択しなければならない。ユニコードからShift-JISにマッピングするときは、コード変換システムは両方のユニコード文字を同一Shift-JIS文字にマッピングすることができ、そのようにするのが通常である。しかし、これらのユニコードからのマッピングの1つだけはユニコードへのマッピングとは逆になっている。

厳格マッピングと緩和マッピングの比較例

- ・ ユニコードu000DがASCIIx0D「キャリッジリターン」に厳格にマッピングされていれば、ユニコードu2029「パラグラフ・セパレータ」はASCIIx0Dにゆるやかにマッピングすることができる。
- ・ ユニコードu002D「ハイフン・マイナス」がASCIIx2D「ハイフン・マイナス」に厳格にマッピングされていれば、ユニコードu2010「ハイフン」とu2212「マイナス記号」はASCIIx2Dにゆるやかにマッピングすることができる。
- ・ ユニコードu00E0「グラーブ付きのラテン小文字A」がISO 8859-1xE0「アクサングラーブ付きの小文字a」に厳格にマッピングされていれば、2文字のユニコード・シーケンスu0061+u0300「ラテン小文字A」+「結合アクサングラーブ」

はISO 8859-1 x EOにゆるやかにマッピングすることができる。

Shift-JISは半幅文字と全幅文字を区別しているので、Shift-JISの緩和マッピングもこれらを区別しておかなければならない。つまり、ユニコードuFF40「全幅アクサングラフ」はShift-JIS x 814D「アクサングラフ（全幅）に厳格にマッピングされており、これはShift-JIS x 60「アクサングラフ（半幅）」と区別されている。ユニコード・シーケンスu3000 + u0300「表意文字スペース」 + 「結合アクサングラフ」はShift-JIS x 814Dにゆるやかにマッピングすることができる。しかし、ユニコード・シーケンスu0020 + u0300「スペース」 + 「結合アクサングラフ」はShift-JIS x 814Dにゆるやかにマッピングしてはならない。これはShift-JIS x 60にマッピングされるべきである。

本発明の第1側面によれば、ユニコードからある種の他の文字にマッピングし再び元に戻すマッピング（ラウンドトリップ・マッピング）は、他の文字への厳格なマッピングが存在しているユニコード文字だけを使用するとき可能である。

さらに、本発明の第1から第5までの側面によれば、フォールバック・マッピングはユニコード文字の意味または同一性を保存していないユニコードからのマッピングである。つまり、これらはユニコード文字（または文字シーケンス）をその定義または用途がユニコード文字の意味または用途を含んでいないターゲット文字セット内の文字（または文字シーケンス）にマッピングする。これに対して、フォールバック・マッピングによると、ユニコード文字（または文字シーケンス）に最も近く対応しているターゲット・コード化内の文字（または文字シーケンス）が得られる。

フォールバック・マッピングの例

- ・ ユニコード文字u0300「結合アクサングラフ」はフォールバック・マッピングとしてASCII x 60「アクサングラフ [スペース]」に

マッピングすることができる。違いは、ユニコード文字が結合マーク（非スペース）であるのに対し、ASCII文字はスペース・マークであることである。

- ・ ユニコード文字 u 0 1 C 0 「ラテン文字デンタル・クリック」はフォールバック・マッピングとして ASCII x 7 C 「垂直線」にマッピングすることが可能である。
- ・ ユニコード文字 u 2 0 0 1 「EM QUAD」はフォールバック・マッピングとして ASCII x 2 0 「スペース」にマッピングすることが可能である。

従って、上記の例に示すように、フォールバック・マッピングはユニコード文字（またはシーケンス）にグラフィックが近似しているターゲット文字（またはシーケンス）を生成するために使用される。

パフォーマンス上の理由により（つまり、コード化をマッピング・テーブルから得るときの速度）、マッピング・テーブル 4 1 4 までのインデックスにはいくつかのフォーマットがある。可能とされるフォーマットには、セグメント・フォーマット、リスト・フォーマット、範囲フォーマット、チェーン・フォーマットがある。文字シーケンスの長さが異なるごとに、別々のインデックスを設けることが好ましい。その結果として、セグメント 5 0 2、5 0 4、5 0 6 の各々に関連するインデックスは異なるフォーマットにすることができ、各インデックスの先頭の情報はそのフォーマットを指定している。フォーマットに関係なく、各インデックスは最終的には、入力シーケンスを直接に出力シーケンスにマッピングするか、あるいは出力シーケンスが長ければ、対応する出力シーケンスのロケーションを指定しているオフセットにマッピングする。

マッピング・テーブル 4 1 4 までのインデックスのチェーン・フォーマットについては、以下で詳しく説明する。チェーン・フォーマットでは、セクションの先頭がチェックされ、それがチェーン・フォーマット・テーブルのチェーン・ヘッダであるか、他のフォーマットであるかが判断される。チェーン・フォーマットは複数のインデックス・テーブルのチェーンを指定し、各々は異なるフォーマット

ットになっている場合がある。必要とするマッピングが最初のインデックス・テーブルに見つからなければ、ルックアップ・ハンドラ412は2番目のインデックス・テーブルを調べ、以下同様である。チェーン・フォーマットは、例えば、あるインデックス・フォーマット（これは空間的および／または時間的に効率的である）が入力シーケンスのすべてではないが、大部分をマッピングすることができるのに対し、別の非効率なインデックス・フォーマットは少数の残りのシーケンスを処理できるときに利用すると、便利である。チェーン・メカニズムがないと、インデックス・シーケンスのすべてに非効率のフォーマットを使用しなければならないことになる。チェーン・フォーマットは変種と許容レベルが異なるたびに、異なるサブテーブルが必要になるときにも便利である。チェー

ン内の各サブテーブルは、現在取り扱われているマッピング許容範囲と変種に基づいてそれを含めるか、除外させるビット・フラグをもっている。本発明の第2、第3および第4側面によれば、ルックアップ・ハンドラ412がターゲット・コード化のマッピング・テーブル414をサーチするときは、含められたサブテーブルだけが考慮される。

さらに、本発明の第1から第4までの側面によれば、各サブテーブルに関連するこれらのビット・フラグはサブテーブル・マスクを形成している。また、呼び出し側要求（例えば、コード化の変種と許容範囲）と判断された属性（例えば、解明された方向とコンテキスト）は選択マスクを形成している。サブテーブル・マスクと選択マスク内のビット割当ては同じである。従って、特定のサブテーブルを含めるかどうかの判断はサブテーブル・マスクと選択マスクのビットワイズANDをとり、そのあと結果をサブテーブル・マスクと比較することによって行われる。ビットワイズANDの結果がサブテーブルのサブテーブル・マスクと同じであれば、そのサブテーブルが含められる。そうでなければ、それは含められない。

マッピング・テーブル414のヘッダ500は好ましくは次のものを収めている。

・ 一般的識別情報フォーマット、長さ、チェックサムおよびバージョン。

- ・ 最小ターゲット文字サイズ（バイト数）（文字サイズともいう）。
- ・ 一般的フラグ（例えば、そのルックアップ・テーブルが方向またはコンテキスト・データを必要としているかどうか）。
- ・ そのテーブルによって処理される最大入力シーケンス長、および1からこの最大長までの入力シーケンス長を処理するテーブルを指定しているオフセット／長さのペアのリスト。
- ・ そのユニコードからのマッピングのデフォルト・フォールバック文字または文字シーケンス。
- ・ そのテーブルによってサポートされる変種のカウントとリスト。各変種ごとに、1つまたは2つ以上の関連ビット・マスクが指定される。単一の変種に複数のビット・マスクがある場合は、属性情報（方向、コンテキスト

ト、および垂直形式の要求）はどのビット・マスクが使用されるかを判断するために使用される。ビット・マスクで「1」にセットされたビットは異なる変種をサポートするために種々のサブテーブルをオンにするために使用される。

- ・ 可能とされる4つの許容範囲設定値（厳格／緩和、フォールバック・オン／オフ）の各々に関連する追加のビット・マスクのセット。該当する許容範囲レベル・マスクは変種マスクとORがとられて、サブテーブルを使用可能または使用禁止にするために使用されるビット・マスクを形成する。

C. コード変換処理

以下では、ユニコード・コード変換システム400の好適実施例によって実行される処理について詳しく説明する。

図6Aはユニコード・コード変換システム400を利用するアプリケーション（つまり、呼び出し側アプリケーション・プロセスまたはプログラム）によって実行される処理600を示すフローチャートである。具体的には、図6Aはユニコードからの処理に関係しているが、当然に理解されるように、類似のオペレーションを実行して他方の方向に変換することも可能である（ユニコードへの処理

）ユニコードからのコンバータ402は全体的変換プロセスを制御する。

最初に、処理600は新しい状態インスタンスと変換のための制御情報を作成し、初期化する（602）。処理600はインスタンスを設定するので、複数のスキャン・オペレーションを進行中にし、そのインスタンスによって区別することが可能である。次に、切り捨てが必要であるかどうかの判断604が行われる。切り捨てが必要な場合には、トランケータ407が起動される（606）。切り捨てが必要になるのは、入力データ・ストリームが、変換のためのデータを収めている受入れバッファ405の容量を越えたときである。切り捨てが必要でなければ、あるいは切り捨てが必要な場合には、ブロック606に続いて、ユニコードからのコンバータ402がコールされ（608）ユニコード文字列404を変換する。ユニコードからのコンバータ402の機能は、テキスト要素を取得

し、そのためのターゲット・マッピングを調べることである。これについては、以下で詳しく説明する。変換機能から戻ると、処理600はユニコードからのコンバータ402からターゲット文字列406を受け取る。

次に、処理600は変換が失敗したかどうかを判断する。変換が失敗していれば、エラーが処理される（614）。他方、変換が正常に行われていれば、変換が完了したかどうかの判断616が行われる。ユニコード文字列404の文字がターゲット・コード化に変換されたとき変換は完了する。変換が完了していれば、処理600は完了し、ターゲット文字列406はコード変換を要求したプロセスまたはアプリケーションに利用可能にされる。さらに、処理600は状態インスタンスと変換のための制御情報を破棄する（618）。他方、変換がまだ完了していないと判断616されたときは、処理600は変換が完了するか、エラーが起こるまでブロック604～616を繰り返す。

図6Bは本発明の第4側面による切り捨て処理620を示すフローチャートである。切り捨て処理620は図6Aのブロック606で起動され、トランケータ407によって実行される。

切り捨て処理620は出力長をゼロ（0）に初期化する（622）。出力長はバッファの実効長、つまり、バッファの切り捨てられた長さに一致している。次

に、次のテキスト要素が取得される（624）。次のテキスト要素の取得（624）に関連する処理はスキャナ408によって行われるが、これについては図9A～図9Cを参照して以下で詳しく説明する。次に、テキスト要素がバッファの物理的長さ（バッファ長）を越える可能性があるかどうかの判断626が行われる。テキスト要素がバッファ長を越える可能性がなければ、出力長はそのテキスト要素を含むように更新される（628）。ここでは、実効長はテキスト要素がバッファ長を越える可能性がない限り、テキスト要素単位で大きくなる。ブロック628に続いて、考慮すべき追加テキストがバッファに残っているかどうかの判断630が行われる。考慮すべき追加テキストがバッファに残っていれば、処理はループしてブロック624に戻り、切り捨て処理620を繰り返す。他方、考慮すべき追加テキストがバッファに残っていないか、あるいはテキスト要素がバッファ長を越える可能性があるとしてブロック626で判断されていれば、

出力長が戻される（632）。戻された出力長（632）はバッファの実効長（つまり切り捨てられた長さ）であるので、バッファは実効的にテキスト要素の終わり（つまり、バッファ内のテキストの最後のテキスト要素）で終了する。

以上のようにして、切り捨て処理はバッファの実効長（つまり、切り捨てられた長さ）を判断する。これはバッファの切り捨てられた部分とも呼ばれる。切り捨ての後バッファに残っている余分のテキストは残余部分と呼ばれる。残余部分は入力ソース文字列の次のバッファ部分にキャリーオーバーされ、その部分と一緒に考慮される。この処理はテキスト要素が正しく判断されることを保証する。

以下は切り捨て処理の使い方の例である。

例：“... ABCD`EFG...”

バッファに置かれている部分が“D”の直後で終わっていれば、切り捨て処理はバッファに置かれている部分を切り捨てて、切り捨てられた長さが“C”のあとで終わるようにする。バッファに置かれた部分を切り捨てる必要があるのは、そのようにしないと、テキスト要素“D”がそのあとに置かれた結合マーク“`”から切り離されることになるからである。切り離されると、テキストはターゲット・コード化に正しく変換されないことになる。残余部分は“D`”であ

り、次の部分にキャリーオーバーされ、“D`EFG..”となる。

図7はユニコード・コンバータ処理700を示すフローチャートである。ユニコード・コンバータ処理700は図6Aのブロック608で実行されるオペレーションと関連づけられている。

ユニコード・コンバータ処理700は判断702から始まる。この判断702は変換すべきテキストがあるかどうかを判断する。変換すべきテキストがなければ、ユニコード・コンバータ処理700は戻るだけである（または完了する）。他方、変換すべきテキストがあれば（つまり、ユニコード文字列404が完全に処理されていなければ）、処理700が続けられる。まず、オフセットがオフセット配列に対して更新される（704）。オフセット配列入力文字列に関連するオフセット（ポインタ）の配列であり、フォント変更、ライン中断、言語変更

などのある種の変更が、呼び出し側アプリケーションが有意と判断している入力文字列404内のどこで行われたかを示している。オフセット配列の更新704は、異なる文字長に合わせてオフセット（ポインタ）を調節することにより行われる。例えば、入力ユニコード文字列404のユニコード文字は2バイト長であるのに対し、ASCIIのターゲット・コード化に関連する文字のサイズは1バイト長である。ここでは、オフセット配列の更新704はオフセットがターゲット・コード化内の対応する文字を指すようにオフセットを更新する。実際には、入力文字列内のオフセットはコード化が異なっている出力文字列にマッピングさる。次に、次のテキスト要素が取得される（706）。スキャナ408はスキャナ・テーブル410はユニコード文字列404からのテキスト要素を判断する。次のテキスト要素の取得706は以下で詳しく説明する。次に、取得されたテキスト要素はマッピング・テーブル414で調べられ（708）、ターゲット・コード化内のテキスト要素の変換コードが得られる。このルックアップはマッピング・テーブル414を使用してルックアップ・ハンドラ412によって行われる。変換コードのルックアップ708についても以下で詳しく説明する。

次に、テキスト要素の変換コードが見つかったかどうかの判断710が行われる。変換コードが見つかったときは、ユニコード文字列404とターゲット文字

列406の入力位置ポインタと出力位置ポインタがそれぞれ更新される(712)。入力位置ポインタは入力文字列404がどれだけ変換されたかを示している。出力位置ポインタはターゲット文字列406の長さを示している。ブロック712に続いて、処理700はユニコード・コンバータ処理700の先頭に戻り、変換すべきユニコード文字列404の次のテキスト要素(存在する場合)が処理できるようにする。

しかるに、変換コードがマッピング・テーブル414に見つからないと判断710が判断したときは、呼び出し側(例えば、呼び出し側アプリケーション)がフォールバック処理を要求していたかどうかの判断714が行われる。呼び出し側がフォールバック処理を要求していれば、フォールバック処理が実行される(716)。フォールバック処理はフォールバック・ハンドラ416によっ

て実行される。これは以下で詳しく説明する。他方、呼び出し側がフォールバック処理を要求していなければ、テキスト要素がルックアップ・ハンドラ412によってターゲット・コード化に変換できなかったのでエラーが通知される(718)。ブロック716と718に続いて、ユニコード文字列404とターゲット文字列406の入力位置ポインタと出力位置ポインタがそれぞれ更新され(702)、そのあと処理はユニコード・コンバータ処理700の先頭に戻り、変換すべきユニコード文字列404の次のテキスト要素(存在する場合)が処理できるようにする。

図8はオフセット更新処理800を示すフローチャートである。オフセット更新処理800はオフセット配列が更新される図7のブロック704と関連している。

オフセット更新処理800は現在入力位置がオフセット配列にあるかどうかを判断802することから始まる。現在入力位置がオフセット配列にあれば、オフセット配列の内容がブロック804に続いて現在出力位置長さに従って更新される(804)、オフセット更新処理800から戻る。他方、現在入力位置がオフセット配列にないときは、更新すべきオフセットがないのでオフセット更新処理800から戻る判断802が行われるだけである。

図9Aおよび図9Bは本発明の第1、第2および第4側面による次のテキスト要素処理900を示すフローチャートである。次のテキスト要素処理900は次のテキスト要素を取得するとき図7のブロック706で実行されるオペレーションを具体化したものである。好ましくは、次のテキスト要素処理900はスキャナ・テーブル410を使用してスキャナ408によって行われる。

次のテキスト要素処理900は状態と再配列フラグを初期化(902)することから始まる。次に、マッピング・テーブル414が方向情報を必要としているかどうかの判断904が行われる。マッピング・テーブル414が方向情報を必要としていれば、ユニコード文字列404の方向が解明される(906)。本発明の第2側面では、マッピング・テーブル414が方向情報を必要としていれば、ユニコード文字列404の次の入力文字の方向が解明される(906)。方向が解明されると(906)、あるいは方向が必要でない場合には判断904の

後で、文字のコンテキストに基づく判断908が行われる。ユニコード文字列404内の文字のコンテキストがコード変換(マッピング)に影響する可能性があるときは、コンテキストが解明される(910)。ブロック910に続いてまたは判断908に続いて、コンテキストが重要でないときは、ユニコード文字がユニコード文字列404から取得される(912)。ここでは、ユニコード文字列404内の次の文字が取得される。次に、取得されたユニコード文字の属性が調べられる(914)。属性ルックアップ914は図11～図13を参照して以下で詳しく説明する。次に、次のテキスト要素処理900のアクションが判断される(916)。アクションの判断916は図14A、図14Bおよび図15を参照して以下で詳しく説明する。

次に、アクションが“END”であるかどうかの判断918が行われる。アクションが“END”でなければ、アクションは少なくとも“ADVANCE”である。アクションが“ADVANCE”であるときは、アクションが“MARK”でもあるかどうかの判断920が行われる。アクションが“MARK”でもあるときは、文字がテキスト要素に挿入される(922)。さらに、“MARK”アクションがとられると双方向状態がセーブされる(924)。双方向状態は方

向性埋め込みスタックと双方向ステート・マシンの現在状態を含んでいる。ブロック924に続いてまたはアクションが“MARK”でもないときは判断ブロック920に続いて、アクション修飾子に基づいてスイッチ・オペレーションが実行される。アクション修飾子はアクションの一部であり、“S”、“ISS”、“ASS”などの修飾子を含んでいる。アクションはどのアクション修飾子ももたないこともできる。アクション修飾子が“S”であるときは、再配列フラグがセットされる(928)。再配列フラグ(セットされたときは)はテキスト要素内の文字を再配列する必要がないことを示している。アクション修飾子が“ISS”(つまり、シメトリック・スワッピング禁止)であるときは、スワップ・フラグがオフにセットされる(930)。アクション修飾子が“ASS”(つまり、スワッピング禁止活動化)であるときは、スワップ・フラグがオンにセットされる(932)。スワップ・フラグはシメトリック・スワッピングが必要かどうかを示している。スイッチ926は拡張

エリア934を使用して追加のアクション修飾子を含むように容易に適應させることができる。拡張エリア934を使用すると、ユーザはスキャナ408の振舞を変更することができる。また、アクション修飾子がなければ、次のテキスト要素処理900はアクション修飾子に関連するどのオペレーションも実行しない。アクション修飾子オペレーションに続いて、現在文字インデックスが更新される(936)。現在文字インデックスは次のテキスト要素処理900を実行するときソース文字列をスキャンしていくために使用されるソース文字列を指すポインタである。次に、次のテキスト要素処理900はブロック904から始まるオペレーションを繰り返す。処理はアクションが“END”であると判断918が判断するまでループしてブロック904-936を繰り返す。アクションが“END”であると、判断918は判断ブロック938を実行させる。判断ブロック938は再配列フラグがセットされているかどうかを判断する。再配列フラグがセットされていれば(ブロック928)、テキスト要素内の文字は再配列される(940)。再配列は異種の文字クラスの加重値を提供する優先順位属性を使用して行うことが好ましい。再配列フラグがセットされていない場合にはブロック9

38に続いて、再配列フラグがセットされている場合にはブロック940に続いて、次のテキスト要素処理900は完了し、ユニコード・コンバータ処理700に戻る。

図10A、図10Bおよび図10Cは本発明の第3側面による次のテキスト要素処理900を示すフローチャートである。次のテキスト要素処理900は次のテキスト要素を取得するとき図7のブロック706によって実行されるオペレーションを具体化したものである。好ましくは、次のテキスト要素処理900はスキャナ・テーブル410を使用してスキャナ408によって実行される。

次のテキスト要素処理900は状態と再配列フラグを初期化(952)することから始まる。次に、マッピング・テーブル414が方向情報を必要としているかどうかの判断954が行われる。マッピング・テーブル414が方向情報を必要としていれば、ユニコード文字列404の次の入力文字の方向が解明される(956)。次に、ユニコード文字がユニコード文字列414から取得される(958)。ここでは、ユニコード文字列404内の次の文字が取得される

(958)。次に、取得されたユニコード文字の属性が調べられる(960)。属性ルックアップ960は図11～図13を参照して以下で詳しく説明する。次に、次のテキスト要素処理900のアクションと次の状態が判断される(962)。アクションと次の状態の判断は図14A、図14B、図16Aおよび図16Bを参照して以下で詳しく説明する。

次に、アクションが“END”であるかどうかの判断964が行われる。アクションが“END”でなければ、アクションは少なくとも“ADVANCE”である。アクションが“ADVANCE”であるときは、アクションが“MARK”でもあるかどうかの判断966が行われる。アクションが“MARK”でもあるときは、文字がテキスト要素に挿入される(968)。さらに、“MARK”アクションがとられると双方向状態がセーブされる(970)。双方向状態は方向性埋め込みスタックと双方向ステート・マシンの現在状態を含んでいる。ブロック970に続いてまたはアクションが“MARK”でもないときは判断ブロック966に続いて、アクション修飾子に基づいてスイッチ・オペレーション97

2 が実行される。アクション修飾子はアクションの一部であり、“S”、“ISS”、“ASS”などの修飾子を含んでいる。アクションはどのアクション修飾子ももたないこともできる。アクション修飾子が“S”であるときは、再配列フラグがセットされる(974)。再配列フラグ(セットされたときは)はテキスト要素内の文字を再配列する必要がないことを示している。アクション修飾子が“ISS”(つまり、シメトリック・スワッピング禁止)であるときは、スワップ・フラグがオフにセットされる(976)。アクション修飾子が“ASS”(つまり、スワッピング禁止活動化)であるときは、スワップ・フラグがオンにセットされる(980)。スワップ・フラグはシメトリック・スワッピングが必要かどうかを示している。スイッチ972は拡張エリア980を使用して追加のアクション修飾子を含むように容易に適應させることができる。拡張エリア980を使用すると、ユーザはスキャナ408の振舞を変更することができる。また、アクション修飾子があれば、次のテキスト要素処理900はアクション修飾子に関連するどのオペレーションも実行しない。アクション修飾子オペレーションに続いて、現在文字インデッ

クスが更新される(982)。現在文字インデックスは次のテキスト要素処理900を実行するときソース文字列をスキャンしていくために使用されるソース文字列を指すポインタである。次に、次のテキスト要素処理900はブロック954から始まるオペレーションを繰り返し替える。処理はアクションが“END”であると判断964が判断するまでループしてブロック954～982を繰り返す。アクションが“END”であると、判断964はコンテキスト処理984を実行させる。コンテキスト処理984のあと、判断ブロック986は再配列フラグがセットされているかどうかを判断する。再配列フラグがセットされていれば(ブロック974を参照)、テキスト要素内の文字は再配列される(988)。再配列は異種の文字クラスの加重値を提供する優先順位属性を使用して行うことが好ましい。再配列フラグがセットされていない場合にはブロック986に続いて、再配列フラグがセットされている場合にはブロック988に続いて、次のテキスト要素処理900は完了し、ユニコード・コンバータ処理700に戻る。

次に、図9Cを参照してコンテキスト処理984について説明する。コンテキスト処理984はここで説明している実施例では、スキャナ・テーブル410を使用するスキャナ408によって実現されている。コンテキスト処理984は判断985から始まり、アクションがEndOutputXn（つまり、単独コンテキスト）であるかどうか判断される。そうであれば、コンテキスト・マスクがセットされ（986）、コンテキストが単独であることを示し、コンテキスト処理984は完了し、戻る。他方、アクションがEndOutputXnでなければ、アクションがEndOutputXl（つまり、初期コンテキスト）であるかどうか判断987で判断される。そうであれば、コンテキスト・マスクがセットされ（988）、コンテキストが初期であることを示し、コンテキスト処理984は完了し、戻る。他方、アクションがEndOutputXlでなければ、アクションがEndOutputXr（つまり、終了コンテキスト）であるかどうか判断989される。そうであれば、コンテキスト・マスクがセットされ（990）、コンテキストが初期であることを示し、コンテキスト処理984は完了し、戻る。他方、アクションがEndOutputXrでなければ、アクションがEndOutputXm（つまり、中間コンテキスト）であるかどうか

判断991が行われる。そうであれば、コンテキスト・マスクがセットされ（992）、コンテキストが初期であることを示し、コンテキスト処理984は完了し、戻る。アクションがEndOutputXmでないと判断991されたときは、テキスト要素は関連するコンテキストをもっていないので、コンテキスト・マスクは「無視」にセットされる（993）。

以上のように、コンテキスト処理984は上記実施例では、テキスト要素とコンテキストと一緒に判断されるように実現されている。どちらの判断も、スキャナ408とスキャナ・テーブル410によって先読みスキニングを利用してテキスト要素が完成し、入力テキスト・ストリーム内のテキスト要素の内容が分かるようにしている。コンテキスト・マスクに入っているコンテキスト情報は、マッピング・テーブル414とやりとりして、判断されたコンテキストをもつテキスト要素の正しいターゲット・コード化を探すときにルックアップ・ハンドラ

4 1 2によって使用される。コンテキスト処理のオペレーションの詳細は図16 Aと図16 Bを参照して以下で説明する。

図11はスキャナ408を示すブロック図である。特に、スキャナ408は属性ハンドラ1000とテキスト要素ハンドラ1002を含んでいる。テキスト要素ハンドラ1002は図9Aと図9Bおよび図10A～図10Cを参照して上述した次のテキスト要素処理900を実行する。属性ハンドラ1000は属性テーブル1004とやりとりして、次のテキスト要素処理900が必要とするユニコード文字の属性を取得する(図9Aのブロック914と図10Aのブロック960を参照)。属性には次のものがある。つまり、方向、クラス、優先順位、シメトリック・スワッピング、サブセットおよびコンテキストである。方向属性は方向を解明するとき使用される(図9Aのブロック914および図10Aと図17Aのブロック956を参照)。クラス属性はアクション(例えば、ADVANCE、END)を判断するためにスキャナ408によって使用される。優先順位属性はテキスト要素内の文字を記録するために使用される(図9Bのブロック940と図10Aのブロック988を参照)。シメトリック・スワッピング属性はシメトリック・スワッピングが必要であるかどうかを判断するために使用される。コンテキスト属性はコンテキストを解明するとき使用される

(図9Aのブロック910を参照)。

図12は図11の属性テーブル1004の好ましいフォーマットを示す概略図である。属性テーブル1004はヘッダ部分1100、範囲テーブル部分1102、および属性テーブル部分1104を含んでいる。ヘッダ部分1100は次のものに関する情報を含んでいる。つまり、各テーブルの総テーブル・サイズ、チェックサム値、バージョン、オフセットおよび要素の数である。範囲テーブル部分1102内の要素は属性が共通にグループ化されている範囲を示し、各グループごとに、属性テーブル部分1104の該当部分を指すポインタをもっている。属性テーブル1004の範囲テーブル部分1102のフォーマットは、例えば、各エントリごとに、範囲値の開始、範囲値の終了、および範囲に関連するデータ・ワードを含んでいる。属性テーブル1004の構成は属性情報のコンパクトな

ストアを容易にしている。別のストア構成も可能であるが、データ・ストアのコンパクトさから見たとき非効率的である。

図13は属性ルックアップ処理1200を示すフローチャートである。属性ルックアップ処理1200は属性ハンドラ1000によって実行され、図9Aのブロック914または図10Aのブロック960で次のテキスト要素処理900によって開始される。

属性ルックアップ処理1200は属性テーブル1004の範囲テーブル部分1102内の範囲を使用してバイナリ・サーチから開始される。該当の範囲がバイナリ・サーチで見つかり、その範囲に関連するデータ・ワードが範囲テーブル部分1102から取得される(1204)。好ましくは、データ・ワードの最初のビットは間接ビットである。次に、データ・ワードの間接ビットがセットされているかどうかの判断1206が行われる。データ・ワードの間接ビットがセットされていない場合は、データ・ワード自体は現在の文字の属性を収めているので、データ・ワードは属性として戻される(1208)。他方、データ・ワードの間接ビットがセットされていれば、属性は、範囲テーブル部分1102から取得されたデータ・ワードを属性テーブル部分1104までのインデックスまたはオフセットとして使用して属性テーブル部分1104から取得される。従って、この場合のデータ・ワードは範囲内の各文字の属性を収めている配列ま

でのインデックスまたはオフセットである。ブロック1208または1210に続いて、属性ルックアップ処理1200は完了し、戻る。

図14Aと図14Bは次のアクションを判断するためにスキャナ408によって使用されるスキャナ・テーブル1300(410)に関連する概略図である。次のアクションの判断は次のテキスト要素処理900(図9A)のブロック916によって起動されるか、ブロック962(図10A)によって起動される。図14Aは本発明で使用するスキャナ・テーブル1300の好ましいフォーマットを示す図である。スキャナ・テーブル1300は「現在状態」を1つのインデックスとして、「クラス」を別のインデックスとしてもつ2次元配列である。「クラス」はクラス属性を指している。これらのインデックスはスキャナ・テーブ

ル1300内の代表的な要素1302を選択する。図14Bはスキャナ・テーブル1300の代用的な要素1302を示す図である。代表的な要素1302はスキャナ408の次の状態を収めている次の状態部分1304を含み、アクション部分1306はスキャナ408のアクションを示している。実際には、スキャナ408はスキャナ・テーブル1300と一緒にあって、スキャナをどのように動作させるかを判断するステート・マシンを実現している。

スキャナ・テーブル1300は異種の文字コード化に関する異なる次の状態とアクションを収めている。

本発明の第1、第2および第4側面によれば、図15は好ましいレイアウトとスキャナ・テーブル1300にストアされる情報の両方を示しているテーブル1400である。このテーブルに関連する表記は次の通りである。

文字クラス：

CC	－制御文字
OS	－他のスペース
NS	－非スペース
LD	－ラテン・ディジット
FS	－フラクション・スラッシュ
JL(f)	－ジャモス先頭子音（フィラー）
JV(f)	－ジャモス母音（フィラー）
JT	－ジャモス・トレーラ
NU	－有効なユニコード文字でない
ISS	－シメトリック・スワッピング禁止
ASS	－シメトリック・スワッピング活動化

次の状態：

状態0　－終了。テキスト要素を戻すべきかどうかを、2重および半分音符号の状態に基づいて判断する。

状態1　－開始状態

状態2　－非スペース（分音符号）の追加

状態3 - 数値フラクションの有無チェック

状態7 - 朝鮮語ジャモス

アクション:

A d v - [ADVANCE] 次の文字へ進む (現在文字は現在テキスト要素 (TE) に含まれている場合と含まれていない場合がある)

A d v M a r k - [ADVANCE+MARK] 現在文字に最終文字のマークを付け次の文字へ進む

A d v M a r k S - [ADVANCE+MARK+S] 現在文字に最終文字のマークを付け次の文字へ進み、再配列フラグをセット

する

A d v M a r k A S S - [ADVANCE+MARK+ASS] 現在文字に最終文字のマークを付け次の文字へ進み、シメトリック・マッピングを活動化する

A d v M a r k I S S - [ADVANCE+MARK+ISS] 現在文字に最終文字のマークを付け次の文字へ進み、シメトリック・マッピングを禁止する

E n d - テキスト要素を最終のマークを付けた文字で終了する

注意: すべての機能は再配列フラグがセットされているかをチェックして確かめ、非スペース文字を開始ポインタから始めて再配列する。再配列する必要のある非スペース・マークのセットは複数存在することがあるので、文字列全体をチェックするのが最良の方法である。もちろん、文字を再配列すると、再配列フラグはクリアされる。

本発明の第3側面によれば、図16Aと図16Bは好ましいレイアウトと情報の両方を示すテーブル1400であり、情報はソース文字列内のテキスト要素とそのコンテキストの両方の判断を容易にするためにスキャナ・テーブル410にストアされるものである。このテーブルに関連する表記は次のとおりである。

文字クラス:

CC	－制御文字
OS	－他のスペース
NS	－非スペース
LD	－ラテン・ディジット
FS	－フラクション・スラッシュ
DD	－二重分音符号
HD	－半分音符号
CH	－ジャモス先頭子音（フィラー）
JO	－ジャモス母音（フィラー）
JV	－ジャモス子音トレーラ
NU	－有効なユニコード文字でない
ISS	－シメトリック・マッピング禁止
ASS	－シメトリック・マッピング活動化
HH	－高ハーフゾーン
LH	－低ハーフゾーン
V	－V i r a m a
ZWNJ	－ゼロ幅の非ジョインダ
R	－右リンク
D	－2重リンク
C	－リンクを引き起こす
T	－透過

次の状態：

状態0	－終了。テキスト要素を戻すべきかどうかを2重分音符号と半分音符号に基づいて判断する
状態1	－開始状態
状態2	－非スペース（分音符号）追加
状態3	－ラテン・ディジット
状態4	－ラテン・ディジット・シーケンス

- A d v M a r k - [A D V A N C E + M A R K] 現在文字に最終文字の
マークを付け次の文字へ進む

A d v M a r k S - [ADVANCE+MARK+S] 現在文字に最終文字のマークを付け次の文字へ進み、再配列フラグをセットする

A d v M a r k A S S - [ADVANCE+MARK+ASS] 現在文字に最終文字のマークを付け次の文字へ進み、シメトリック・マッピングを活動化する

A d v M a r k I S S - [ADVANCE+MARK+ISS] 現在文字に最終文字のマークを付け次の文字へ進み、シメトリック・マッピングを禁止する

E n d - テキスト要素を最終のマークを付けた文字で終了する

E n d O u t p u t X n - [END+単独コンテキスト] テキスト要素を終了し、単独コンテキストであることを示す

E n d O u t p u t X l - [END+初期コンテキスト] テキスト要素を終了し、初期コンテキストであることを示す

E n d O u t p u t X r - [END+終了コンテキスト] テキスト要素を終了し、終了コンテキストであることを示す

E n d O u t p u t X m - [END+中間コンテキスト] テキスト要素を終了し、中間コンテキストであることを示す

注意：すべての機能は再配列フラグがセットされているかをチェックして確かめ、非スペース文字を開始ポインタから始めて再配列する。再配列する必要のある非スペース・マークのセットは複数存在することがあるので、文字列全体をチェックするのが最良の方法である。もちろん、文字を再配列すると、再配列フラグはクリアされる。

以下では、スキャナ・テーブル1400の使用例を3つ示して説明する。最初の2つの例は本発明の第1、第2、第3および第4側面に関係し、最後の例は第3側面に関係するものである。

例1：入力文字列“AA B”

文字クラスは3文字ともOSである。最初の文字“A”が取得される。開始

状態（状態1）から始まり、最初のアクションはA d v M a r k、次の状態は状態2である。これにより、最初の文字“A”が現在テキスト要素内に挿入され、次の文字（2番目の文字“A”）が取得される。次に、状態2では、アクションはE n d、次の状態は状態0である。従って、テキスト要素は最初のテキスト要素だけを含んでいる。同じシーケンスはこの特定入力文字列の2番目と3番目の文字について繰り返される。このようにして、入力

文字列の文字の各々は分離しているが、隣接しているテキスト要素に割り当てられる。

例2：入力文字列“A ` B”

文字クラスは入力文字列の最初と最後の文字ではO Sである。2番目の文字の文字クラスがN Sであるのは、これが結合マークであるためである。最初の文字“A”が取得される。開始状態（状態1）から始まり、最初のアクションはA d v M a r k、次の状態は状態2である。これにより、最初の文字“A”が現在テキスト要素内に挿入され、次の文字（2番目の文字“`”）が取得される。次に、状態2では、アクションはA d v M a r k S、次の状態は状態2である。これにより、2番目の文字“`”が現在テキスト要素に挿入される。次に、3番目の文字が取得される。この時点の状態2のアクションはE n d、次の状態は状態0である。従って、テキスト要素は入力文字列の1番目と2番目の文字を含んでいる。3番目の文字は例1の場合と同じように、自身のテキスト要素に置かれる。

例3：入力文字列“O S R D O S O S” [文字間にスペースがなく、各文字はその文字クラスで表されている。RとD文字クラスはコンテキスト・ベースの表示形態をもつ文字を含んでいるが、O S文字クラスは含んでいない。]

最初の文字は文字クラスO Sである。開始状態（状態1）から始まり、最初のアクションはA d v M a r k、次の状態は2、次の開始状態は1である。これにより、最初の文字が現在テキスト要素に挿入され、2番目の文字が取得される。2番目の文字は文字クラスRである。次に、状態2では、アクションはE n d、次の状態は0、次の開始状態は1である。従って、最初のテキスト要素は最初

の文字だけを含み、コンテキスト・フラグは無視にセットされる。

次のテキスト要素を判断するときは、処理は2番目の文字から始まり、新しい開始状態は1になっている。この時点で、スキャナ・テーブルから得ら

れるアクションはA d v M a r kであり、次の状態は17、次の開始状態は13である。これにより、2番目の文字が現在テキスト要素に挿入され、3番目の文字が取得される。3番目の文字は文字クラスRである。次に、状態17では、アクションはE n d O u t p u t X n、次の状態は0、次の開始状態は1である。従って、2番目のテキスト要素は2番目の文字だけを含み、コンテキストはX n（単独）である。

次のテキスト要素を判断するときは、処理は3番目の文字から始まり、新しい開始状態は13になっている。この新しい開始状態は最後の文字についてテーブルで示された次の開始状態である。ここでは、スキャナ・テーブルから得られるアクションはA d v M a r kであり、次の状態は15、次の開始状態は13である。3番目の文字は現在テキスト要素の一部であるので、4番目の文字が取得される。4番目の文字は文字クラスO Sである。次に、状態15では、アクションはE n d O u t p u t X r、次の状態は0、次の開始状態は1である。従って、3番目のテキスト要素は3番目の文字だけを含み、コンテキストはX r（終了）である。

次のテキスト要素を判断するときは、処理は4番目の文字から始まり、新しい開始状態は13になっている。スキャナ・テーブルから得られるアクションはA d v M a r kであり、次の状態は2、次の開始状態は1である。これにより、4番目の文字が現在テキスト要素に挿入され、5番目の文字が取得される。5番目の文字は文字クラスO Sである。次に、状態2では、アクションはE n d、次の状態は0、次の開始状態は1である。従って、4番目のテキスト要素は4番目の文字だけを含み、コンテキスト・フラグは無視にセットされる。

図17Aは本発明の第2側面による方向解明処理1500を示すフローチャートである。方向解明処理1500は次のテキスト要素処理900（図9A）内のブロック906で実行される処理である。方向解明処理1500は、スキャナ4

08のために解明された方向の状態が初期状態にあるかどうかを判断する判断1502から開始される。初期状態では、スキャナ408はテキスト要

素内の文字の方向を知らない。スキャナ408のために解明された方向の状態が初期状態にあれば、入力文字列の初期方向が判断される(1504)。初期状態1504の判断1504に関連する処理は図18を参照して以下で詳しく説明する。他方、スキャナ408のために解明された方向の状態が初期状態にな説ときは、初期方向を判断する必要はない。どちらの場合も、ブロック1502またはブロック1504に続いて、ユニコード文字が入力ストリームから取得される(1506)。次に、ユニコード文字に関連する属性が調べられる(1508)。方向属性は方向解明処理1500を行うとき重要な属性である。方向解明処理1500のブロック1506と1508が実行するオペレーションは次のテキスト要素処理900のブロック912と914のそれと同じであるので、これ以上詳しく説明することは省略する。

次に、方向、次の状態およびアクションが判断される(1510)。方向、次の状態およびアクションは方向属性と現在状態を使用して判断され、テーブル属性ルップアップ・プロセスを使用して方向テーブルから取得される。方向テーブルは方向属性と現在状態によってインデックスされる2次元配列である。インデックスが指している方向テーブル内の要素は文字の方向、次の状態、および方向解明処理1500がとるべきアクションを含んでいる。方向は次のものの1つである。すなわち、左、右、グローバル、およびNO__OUTPUTである。取得するアクションは、NO__ACTION、PUSH RO、PUSH RE、PUSH LE、PUSH LO、POP、およびRESETである。方向は明示的オーバーライド文字により入力文字列の中で変化することがあるので、以前の方向は方向スタック上にストアされている。従って、「プッシュ」および「ポップ」の使用はこの分野で周知であるスタック操作コマンドを意味する。“RO”は右から左へのオーバーライドを意味し、“LO”は左から右へのオーバーライドを意味し、“RE”は右から左への埋め込みを意味し、“LE”は左から右への埋め込みを意味する。好ましくは、方向テーブルと方向解明処理1500はステート

・マシンとして動作する。ステート・マシンは基本的にユニコード標準バージョン1.0 611-621ページ（付録A）に記載されている双方向アルゴリズムに従って動作するが、1回のパスだけで結果が得られるのに対し、ユニ

コード標準に記載されているアルゴリズムは複数のパスを必要とする。

図17B～図17Dは本発明の第2側面による双方向アルゴリズムの好ましいインプリメンテーションによる双方向状態テーブル1511を示している。双方向状態テーブル1511はテーブル駆動型ステート・マシンを実現している。各カラムは単一状態である。状態はそこに記録される情報を示唆する名前が付けられている。各行は図17B～図17Dにまたがっており、以下に示す文字クラス名の1つが付けられている。

L R	左から右へが主流
R L	右から左へが主流
A L	アラビア文字（右から左へが主流）
L R E	左から右への埋め込みマーク
R L E	右から左への埋め込みマーク
L R O	左から右へのオーバーライド・マーク
R L O	右から左へのオーバーライド・マーク
P D F	ポップ方向フォーマット・マーク
A N	アラビア数字
E N	ヨーロッパ数字
E T	ヨーロッパ数字ターミネータ
E S	ヨーロッパ数字セパレータ
C S	共通数字セパレータ
O N	その他の中立文字
B S	ブロック・セパレータ

双方向状態テーブル1511の各セルは関連のアクションと出力を持つ遷移を表している。これらのセルは新しい状態の名前、とられるオプションのアクション、および出力（もしあれば）を収めている。新しい状態は現在のグローバル方向

に左右される場合がある。これは新しい状態名に（G）を入れることで示される。グローバル方向が未知のときこれらの遷移の1つを行うとエラーになる。取り得るアクションは次のとおりである。

プッシュ 新しい埋め込み状態を埋め込みスタック上にプッシュする。プッシュされる実際の値は現れた実際の埋め込み制御によって決まる。このインプリメンテーションでは、これ进行处理するアクション動詞は4つある。

ポップ 現在の埋め込み状態をスタックからポップし、スタックの新しいトップを現在の埋め込み状態にする。新しい埋め込みがオーバーライドであれば、セルに入っているターゲット状態ではなくOR状態に遷移が行われる。

リセット 埋め込み状態をクリアし、文字を消費することなくSTARTに移る。すなわち、エプシロン遷移を行う。リセットには出力はない。

エラー 即時エラーを引き起こす。

出力はそれぞれ左から右へと右から左へを意味するLまたはRのどちらか、現在のグローバル方法の出力を意味するG、または出力なしを意味する*である。マシンには、小文字の's'で始まらない名前を持つ状態で入ることができる。START状態は新しいテキスト・ブロックを目的としている。sDIR状態はグローバル方向を判断するときのエントリ・ポイントとして使用される。この計算はメイン・スキャンと同時に行うことが可能であるが、別々にすると単純化される。

図15Aに戻って説明すると、ブロック510に続いて、アクションが“RESET”であるかどうかの判断1512が行われる。アクションが“RESET”であれば、処理は方向解明処理1500の先頭に戻る。そうでなければ、方向解明処理1500が続行される。つまり、判断1512に続いて、アクションが実行される（1514）。

次に、方向（ブロック1510で判断されたもの）が“NO__OUTPUT”

であるかどうかの判断1516が行われる。方向が“NO__OUTPUT”に等しくなければ、方向がコンテキスト（各インスタンスごとに）にセットされ（1518）、状態が保存される（1520）。ブロック1520に続いて、方

向解明処理1500は完了し、戻る。

しかるに、出力が“NO__OUTPUT”であると判断1516で判断されたときは、入力文字列全体が処理されたかどうかの判断1522が行われる。入力文字列全体が処理されていないければ、処理は方向解明処理1500の先頭に戻り追加のユニコード文字を取得し、処理できるようにするが、これはテキスト要素の方向がまだ判断されていないためである。他方、入力文字列全体が処理されたと判断1522されたときは、処理がテキストの終わりまで達したかどうか判断1524される。つまり、ターゲット・コード化に変換すべきテキストが方向解明処理1500によって完全に処理されたかどうか判断される。変換すべき追加テキストがあれば、入力文字列の現在の文字では方向が計算できなかったのがエラーになる（1528）。しかし、追加テキストがなければ、ブロック・セパレータの方向がブロックの全体方向から判断される（1526）（1504を参照）。ブロック1526に続いて、ブロック1518と1520が前述したように実行され、そのあと方向解明処理1500は完了し、戻る。

図18は本発明の第2側面による初期方向判断処理1600を示すフローチャートである。初期方向判断処理1600は図17Aのブロック1504で実行されるオペレーションと関連している。

初期方向判断処理1600は制御フラグに基づくスイッチ・オペレーションから開始される。制御フラグは次の1つを示している。すなわち、NO__OUTPUT、L-t-o-o-RまたはR-t-o-o-Lである。これらの制御フラグはユニコード・コード変換システム400を起動するアプリケーションによってセットされる（つまり、制御フラグはコンバータへの入力である）。制御フラグがR-t-o-o-Lを示しているときは、グローバル方向はR-t-o-o-Lにセットされる（1604）。制御フラグがL-t-o-o-Rにセットされているときは、グローバル方向はL-t-o-o-Rにセットされる（1606）。制御フラグがNO__OUTPUT

にセットされているときは、スモール・ループが開始され、方向が判断できるまで入力文字列のユニコード文字をスキャンしていく。ループは状態を“START STATE”にセット（1608）することから開始される。次に、ユニコード文字が入力文字列から取得される（1610）。ユニ

コード文字の属性が次に調べられる（1612）。属性は図9Aのブロック914で使用され、図12に詳しく説明されているものと同じ方法を用いて調べられる（1612）。次にユニコード文字の方向が属性（つまり、方向属性）を使用して判断される（1614）。方向が“NO__OUTPUT”に等しいかどうかの判断1616が行われる。方向が“NO__OUTPUT”であれば、入力文字列の終わりまで達したかどうかの判断1618が行われる。入力文字列の終わりまで達していなければ、処理は戻り、ブロック1610～1618を繰り返す。文字列の終わりまで達していると、判断1618は方向を見つけることなく特殊方向スキヤニング・ループを終わらせる（例えば、NO__OUTPUT）。そうでなければ、スキヤニング・ループは判断1616が方向を判断したとき終了する。

いずれの場合も、スイッチ・オペレーション1602に関連する方向処理に続いて、スイッチ・オペレーション1620は判断に基づいて起動される。判断された方向が2NO__OUTPUTであるときは、現在レベルは埋め込みボトムにセットされる（1622）。他方、方向がL-to-Rであれば、現在レベルはゼロにセットされ、前のレベルはボトムにセットされ、オーバーライド状況は中立にセットされる（1624）。方向がR-to-Lであるときは、現在レベルは1にセットされ、前のレベルはボトムにセットされ、オーバーライド状況は中立にセットされる（1626）。レベルとは、ユニコード標準に記載されている埋め込みレベルのことである。初期方向判断処理1600の特殊方向スキヤニング・ループ（例えば、1610～1618）は図18に別ループとして示されているが、総システム処理効率に特殊方向スキヤニング・ループをメイン方向スキヤニング・ループ内に組み入れると向上することができる。

図19はテキスト要素ルックアップ処理1630を示すフローチャートである

。テキスト要素ルックアップ処理1630はルックアップ・ハンドラ412によって実行され、ユニコード・コンバータ処理700（図7）内のブロック708によって起動される。

テキスト要素ルックアップ処理1630は変種リストをサーチし、実際の属性と要求された変種に一致するエントリを見つけることから開始される。変種の

サーチ1632は図24を参照して以下で詳しく説明する。次に、一致するものが見つかったかどうかの判断1634が行われる。一致するものが見つからなければ、エラーになる（1636）。他方、一致するものが見つければ、対応するビット・マスクが変種リストから取得される（1638）。好ましくは、変種リストは3つのフィールド、つまり、変種識別子、属性のセット、およびビット・マスクをもっている。変種リスト内の変種識別子と属性セットが実際の属性および要求された変種と一致していれば、対応するビット・マスクが変種リストから選択される。ブロック1638に続いて、ビット・マスクは許容範囲ビット・マスクと結合され（1640）選択フラグが得られる。選択フラグは上述したようにマッピング・テーブル414のサブテーブルを選択するとき使用される選択マスクを形成する。好ましくは、本発明の第2側面では、結合1640はビットワイズ・オペレーションである。好ましくは、本発明の第1、第3および第4側面では、結合1640はビットワイズORオペレーションである。次に、現在テキスト要素の長さのためのテーブルがマッピング・テーブル414内にあるかどうかの判断1642が行われる。なければ、エラーになる（1644）。他方、現在テキスト要素の長さのためにテーブルがあれば、ルックアップ・テーブルとそのフォーマットが現在テキスト要素の長さのために取得される（1646）。次に、スイッチ・オペレーション1648がフォーマットに基づいて実行される。このインプリメンテーションで利用できるフォーマットはリスト、セグメント配列、範囲およびチェーンである。フォーマットがリスト・フォーマットであれば、リスト・フォーマット処理1650が実行される。フォーマットがセグメント配列であれば、セグメント配列フォーマット処理1652が実行される。フォーマットが範囲であれば、範囲フォーマット処理1654が実行される。フォー

ットがチェーンであれば、チェーン・フォーマット処理1656が実行される。ブロック1650-1656に続いて、結果が戻され(1658)、これによりテキスト要素ルックアップ処理1630は完了する。

本発明の第1側面によれば、図20はチェーン・フォーマット処理1660を示すフローチャートである。チェーン・フォーマット処理1660は図19に示すチェーン・フォーマット処理1635によって実行される処理である。

チェーン・フォーマット処理1660はチェーン内のテーブル数のチェーン・カウントを取得する(1662)。そのあと、現在のカウントはゼロにセットされる(1664)。次に、現在のカウントがチェーン・カウントより大であるか等しいかどうかの判断1666が行われる。現在のカウントがチェーン・カウントより大または等しければ、チェーン・フォーマット処理1660は戻り(1668)、結果が見つからなかったのでエラーを通知する。他方、現在のカウントがチェーン・カウントより大でも等しくもなければ、現在のルックアップ・テーブルとそのフォーマットが取得される(1670)。このインプリメンテーションで使用されるフォーマットは図19で使用されたものと同じである。そのあとスイッチ・オペレーション1672がフォーマットに基づいて実行される。フォーマットがリストであれば、リスト・フォーマット処理1674が実行される。フォーマットがセグメント配列であれば、セグメント配列フォーマット処理1676が実行される。フォーマットが範囲であれば、範囲フォーマット処理1680が実行される。フォーマットがチェーンであれば、チェーン・フォーマット処理1682が実行される。ブロック1614~1622に続いて、現在のカウントがインクリメントされる(1684)。次に、結果が見つかったかどうかの判断1686が行われる。結果が見つからなければ、チェーン・フォーマット処理1660はループして判断ブロック1666に戻り、テーブル・チェーン内の次のルックアップ・テーブルを調べていく。しかるに、結果が見つかったと判断1686されたときは、結果が戻され(1688)、これによりチェーン・フォーマット処理1660は完了する。

本発明の第1側面によれば、図21は範囲フォーマット処理1700を示すフ

ローチャートである。範囲フォーマット処理1700は図19のブロック1654と巢20のブロック1678によって実行される処理である。範囲フォーマットはデルタ値が各フィールドと関連づけられている文字の範囲リストである。

範囲フォーマット処理1700はそのサブテーブルのサブセット・フラグが選択フラグに一致しているかどうかを判断(1702)することから始まる。選択フラグは選択マスクに対するもので、サブセット・フラグはサブテーブル・マ

スクに対するものである。一致していなければ、範囲フォーマット処理は戻り(1704)、結果が見つからなかったことをエラー・コードで通知する。他方、そのサブテーブルのサブセット・フラグが選択フラグに一致していることを判断1702が示していれば、テキスト要素の長さが1より大であるかどうかの判断1706が行われる。テキスト要素の長さが1より大であれば、このフォーマットは誤って選択されたことになる。この特定インプリメンテーションでのマッピング・テーブルの構成は範囲フォーマットが長さ1のテキスト要素だけを目的とするようになっているためである。従って、テキスト要素が1より大であれば、ブロック1704が実行されて戻り、結果が見つからなかったことを通知する。他方、テキスト要素の長さが1より大でなければ、範囲フォーマット処理1700は続行される。範囲フォーマットをもつサブテーブルは範囲配列をもち、各範囲はそれぞれに関連づけられたデルタ値をもっている。次に、範囲配列がサーチされ(1708)、変換されるユニコード文字の該当する範囲が見つけられる。範囲が見つかったかどうかの判断1710が行われる。範囲が見つからなければ、範囲フォーマット処理1700は戻り(1704)、結果が見つからなかったことをエラー・コードで通知する。しかるに、結果が見つかったときは、その範囲の対応するデルタ値が取得される(1712)。次に、このデルタ値はユニコード値に加えられる(1714)。そのあと、その結果は出力シーケンスにマッピングされる(1716)。結果を出力シーケンスにマッピングする処理は図22A~図22Cを参照して以下で詳しく説明する。ブロック1716に続いて、範囲フォーマット処理は完了し、戻る。

本発明の第1側面によれば、図22はリスト・フォーマット処理1800を示

すフローチャートである。リスト・フォーマット処理1800は図19のブロック1650と図20のブロック1674によって実行される処理である。リスト・フォーマットはテキスト要素が配列されたリストであり、この配列リストまでのインデックス*i*は対応するルックアップ・ターゲット・リストまでのインデックスである。

リスト・フォーマット処理1800はそのサブテーブルのサブセット・フラグが選択フラグに一致しているかどうかの判断1802から始まる。一致していな

ければ、リスト・フォーマット処理1800は戻り(1804)、マッピングが見つからなかったことをエラー・コードで通知する。他方、そのサブテーブルのサブセットフラグが選択フラグに一致していれば、リスト内のテキスト要素に基づいて最適化バイナリ・サーチが行われる(1806)。次に、サーチがリスト内のテキスト要素を見つけたかどうかの判断1808が行われる。見つからなければ、再びリスト処理は戻り(1804)、マッピングが見つからなかったことを通知する。しかし、テキスト要素が見つければ、テキスト要素が見つかった個所のインデックス*i*が取得される(1810)。このインデックス*i*はルックアップ・ターゲットを取得する(1812)ために使用される。次に、ルックアップ・ターゲットは出力シーケンスにマッピングされる(1814)。これでリスト・フォーマット処理は完了し、戻る。

本発明の第1側面によれば、図23Aと図23Bはセグメント配列フォーマット処理1900を示している。セグメント配列フォーマット処理1900は図19のブロック1652と図20のブロック1676によって実行される処理である。セグメント配列フォーマットは第1テキスト要素配列、最終テキスト要素配列、および*n*個のオフセット配列を含んでいる。オフセット配列内のオフセットは種々のルックアップ・ターゲット・リストを指している。

セグメント配列フォーマット処理1900はそのサブテーブルのサブセット・フラグが選択フラグに一致しているかどうかの判断1901から始まる。サブセット・フラグが選択フラグに一致していれば、最適化サーチが行われる(1902)。最適化サーチは最終テキスト要素配列内にあって、探索していたテキスト

要素よりも大であるか等しい最小エントリを見つけ、そのエントリのインデックス i を取得する。次に、第1テキスト要素配列内の i 番目のエントリが探索していたテキスト要素より小であるか等しいかどうかの判断1904が行われる。そうでなければ、セグメント配列フォーマット処理1900は戻り(1906)、マッピングが見つからなかったことをエラー・コードで知らせる。また、判断1901が失敗した場合には、ブロック1906も実行される。他方、最初のテキスト要素配列内の i 番目のエントリが探索していたテキスト要素より小であるか等しいことを判断1904が示しているときは、 i 番目のエン

トリは探索していたテキスト要素に対応することが分かる。そのあと、ルックアップ・ターゲット・リストがオフセット配列内の i 番目のエントリを通して取得される(1908)。すなわち、オフセット配列内の i 番目のエントリに入っているオフセットはルックアップ・ターゲット・リストを示している。ルックアップ・ターゲット・リストまでのインデックス j が次に判断される。インデックス j は探索していたテキスト要素の値から、取得(1908)したルックアップ・ターゲット・リスト(または範囲)内の第1テキスト要素を引いた値が与えられる。インデックス j をもつルックアップ・ターゲットが次にルックアップ・ターゲット・リストから取得される(1912)。次に、ルックアップ・ターゲットがゼロに等しいかどうかの判断1914が行われる。ルックアップ・ターゲットがゼロに等しければ、セグメント配列フォーマット処理は戻り、マッピングが見つからなかったことをエラー・コードで通知する。他方、ルックアップ・ターゲットがゼロに等しくなければ、ルックアップ・ターゲットは出力シーケンスにマッピングされる(1918)。ブロック1918に続いて、セグメント配列フォーマット処理1900は完了し、戻る。

図24はサーチ変種リスト処理2000を示すフローチャートである。サーチ変種リスト処理2000は図19のブロック1632によって実行される処理である。言い換えれば、サーチ変種リスト処理2000はマッピング・テーブル414を使用してルックアップ・ハンドラ412によって実行されるルックアップ・テキスト要素処理1630の一部である。

サーチ変種リスト処理2000は変種リスト内の要素の総カウントを取得する(2002)。そのあと、現在のカウントがゼロに初期化される(2004)。現在のカウントが総カウントより大であるか等しいかどうかの判断2006が行われる。現在のカウントが総カウントより大または等しければ、サーチ変種リスト処理2000は戻り(2008)、変種が見つからなかったことをエラー・コードで通知する。他方、現在のカウントが総カウントより大または等しくなければ、変種リスト内にあって、現在のカウントに関連するエントリが実際の属性および要求された変種に一致しているかどうかの判断2010が行われる。一致していれば、変種リスト内のエントリからの変種フラグが戻される(2012)。一致していなければ、現在のカウントはインクリメントされ(2014)、そのあと処理はブロック2006に戻り、変種の1つが一致するか、あるいは変種のすべてが考慮されるまでループを続けて変種リスト内の使用可能な変種を見つける。

図25Aおよび図25Bは変種リスト2100を示す概略図である。図25Aに示すように、変種リスト2100は変種領域2102、所望属性領域2104および変種フラグ領域2106を含んでいる。図25Bは好ましいインプリメンテーションによる実際属性ビット・マスクを示す図である。実際属性ビット・マスク2108は32ビット変数であり、シメトリック・スワッピング状態を示す第1部分2110(ビット0と1)、垂直または水平形式を示す第2部分(ビット2と3)、解明方向を示す第3部分(ビット8と9)、およびコンテキストを示す第4部分(ビット16~19)をもっている。本発明の第3側面によれば、第4部分2116のビットはコンテキスト処理984(図10C)で判断されたコンテキスト・マスクによってセットされる。さらに、本発明の第1ないし第4側面によれば、部分内の各ビットはフラグを表している。ビットはフラグが未知であるか、あるいはセットされていなければ、値が“0”であり、セットされているときは値は“1”になっている。呼び出し側は第2部分2110をセットし、スキャナ408は第1、第2および第4部分2110、2114および2116をセットする。

所望属性ビット・マスクは実際属性ビット・マスクと同じフォーマットである

が、ビットは属性のどれが特定テーブルおよび変種の正しいマッピングを得るために重要であるかに応じてセットされる（これはマッピング・テーブル414の設計によって決まる）。所望属性ビット・マスク内のビットはマッピング判断の際に考慮される各属性には“1”がセットされ、ある部分のすべてのビットが“1”にセットされていれば、属性はマッピング期間には無視される。例えば、ビット0が“1”で、ビット1が“0”ならば、シメトリック・スワッピングはオンであり、マッピングが行われるとき考慮される。他方、ビット0と1が共に“1”であれば、シメトリック・スワッピングは完全に無視される。所望属性ビット・マスクの残りの未使用ビットは“1”にセットされ、必要ならばあとで値を割り当てることができる。以下、所望属性ビット・マスクの例をいくつか示して説明する。

方向が左から右へであり、他の属性はいずれも重要でないとする。そうすると所望属性ビット・マスクはx F F F F F D F Fとなる。これに対して、方向が右から左へであり、シメトリック・スワッピングがオンであるとする、所望属性ビット・マスクはx F F F F F E F Dとなる。方向が右から左へで、シメトリック・スワッピングがオフであれば、所望属性ビット・マスクはx F F F F F E F Eとなる。上記の異種所望属性ビット・マスクの各々によると、異なる変換コードを選択することができる。例えば、ユニコード文字u 0 0 2 8をM a c A r a b i cにマッピングすると、所望属性ビット・マスクx F F F F F D F Fではx 2 8が、所望属性ビット・マスクx F F F F F E F Dではx A 8が、所望属性ビット・マスクx F F F F F E F Eではx A 9が得られる。

本発明の第1側面によれば、図26A、図26Bおよび図26Cはルックアップ・ターゲットの出力シーケンスへのマッピング処理2200を示すフローチャートである。ルックアップ・ターゲットの出力シーケンスへのマッピング処理2200は図22のブロック1814および図23Aのブロック1918に関連している。

ルックアップ・ターゲットの出力シーケンスへのマッピング処理2200は間接が許されるかどうかの判断2202から始まる。間接が許されなければ、ルッ

クアップ・ターゲットは `charsize` の断片で出力シーケンスにコピーされる (2204)。 `charsize` とは、ターゲット・コード化内の文字の最小サイズのことであり、マッピング・テーブル 414 のヘッダ 500 に指定されている。ブロック 2204 に続いて、処理 200 は完了し、結果が見つかったとの通知を出して戻る (2206)。他方、間接が許されると判断 2202 が判断したときは、ルックアップ・ターゲットの上位ビットが 1 に等しいかどうかの判断 2208 が行われる。上位ビットが 1 に等しくなければ、第 1 バイトがヌル出力シーケンス (つまり、長さが 0 の出力シーケンス) を示しているかどうかの判断 2210 が行われる。第 1 バイトが `x7F` を示していれば、ヌル出力シーケンスが示される (2212) (一致するものが見つかったが、文字は出力シーケンスに追加されない)。他方、第 1 バイトがヌル出力シーケンスを示していないときは、 `charsize` の断片が出力シーケンスにコピーされる (2214)。その場合には、第 1 バイトは出力シーケンスの長さを示していることが好ましい。ブロック 2212 と 2214 に続いて、ブロック 2206 が実行され、これにより処理は完了し、結果が見つかったことを通知して戻る。

しかるに、ルックアップ・ターゲットの上位ビットが 1 に等しいと判断 2208 が判断した場合には、ルックアップ・ターゲットは所望の出力シーケンスを間接的に参照しているので追加の処理が必要になる。具体的には、間接シーケンスまでのオフセットがルックアップ・ターゲットの残余部分を使用して指定される (2216)。ルックアップ・ターゲットの残余部分の上位ビットが 1 に等しいかどうかの判断 2218 が行われる。等しくなければ、 `charsize` の断片が出力シーケンスにコピーされ (2220)、そのあと処理 2200 はマッピングが見つかったとの通知を戻して (2200) 完了する。他方、ルックアップ・ターゲットの残余部分の上位ビットが 1 に等しいと判断されていれば (2218)、順次チェーン内のシーケンスのカウントが取得され (2224)、リニア・サーチがシーケンスを通るように行われ、マッピングが特定される。ブロック 2226 に続いて、前述したブロック 2220 と 2222 が実行される。

図 26C は図 26B のブロック 2226 によって実行されるオペレーションの

詳細図である。すでに説明したように、図26Cのブロック2226はシーケンスを通るリニア・サーチを実行する。最初に、現在のカウン트가ゼロにセットされる(2228)。現在のカウン트가シーケンス・カウンタより大であるか均かどうかの判断2228が行われる。現在のカウン트가シーケンス・カウンタより大または等しければ、ルックアップ・ターゲットの出力シーケンスへのマッピング処理2200はマッピングが見つからなかったとの通知と共に戻る(2232)。他方、現在のカウン트가シーケンス・カウンタより大または等しくなければ、次の出力シーケンスとそのシーケンス・マスクが取得される(2234)。シーケンス・マスクは複数の出力シーケンスの1つを選択するために使用されるマスクである。次に、サブセット・フラグが使用中であるかどうかの判断2236が行われる。サブセット・フラグが使用中でなければ、シーケンス・マスクと論理ANDがとられた実際属性が実際属性にビットワイズに等しいかどうかの判断2238が行われる。そうであれば、マッピングが見つかったので処理は上述したブロック2220と2222を実行する。他方、サブセット・フラグが使用中であると判断(2236)されたときは、シーケンス・マスクと論理ANDがとられた選択フラグがビットワイズにシーケンス・マスクと等しいかどうかの判断2240が行われる。そうであれば、マッピングが見つかったので前述したブロック2220と2222が実行される。従って、ブロック2238と2240は異なった方法で正しい出力シーケンスを取得する。他方、どちらかの判断ブロック2238または2240が現在のシーケンスが正しいマッピングでないと示していれば、現在のカウンタはインクリメントされ(2242)、処理はブロック2230に戻ってチェーン内の次のシーケンスのオペレーションを続ける。

図27は本発明によるフォールバック・ハンドリング処理2300を示すフローチャートである。フォールバック・ハンドリング処理2300はフォールバック・ハンドラ416によって実行され、図7に示すユニコード・コンバータ処理700のブロック716によって起動される処理である。

フォールバック・ハンドリング処理2300はフォールバック・オプションを使用してテキスト要素を調べる(2302)。ルックアップ2302は図19を

参照して上述したテキスト要素ルックアップ処理1630に類似している。唯一の重要な違いは、変化した選択フラグを通して追加サブセットが考慮されるようにフォールバック・オプションがセットされていることである。次に、テキスト要素の変換コードが見つかったかどうかの判断2304が行われる。変換コードが見つかっていれば、変換またはマッピングを取得するときにどのフォールバックが使用されたかを示すようにエラー・コードがセットされる(2306)。ブロック2306に続いて、フォールバック・ハンドリング処理2300は完了し、戻る。

他方、判断2304が変換コードが見つからなかったと示しているときは、フォールバック・オプションに基づいてスイッチ・オペレーション2308が実行される。フォールバック・オプションには、次のものがある。すなわち、デフォルト、呼び出し側定義、呼び出し側定義を伴うデフォルト、またはデフォルトを伴う呼び出し側定義である。フォールバック・オプションがデフォルトであれば、スイッチ・オペレーション2308はデフォルト処理2310を実行させる。フォールバック・オプションが呼び出し側定義であれば、スイッチ・オペレーション2308は呼び出し側定義のオペレーションを2312を実行させる。フォールバック・オプションが呼び出し側定義を伴うデフォルトであれば、スイッチ・オペレーション2308はデフォルト処理を実行させ(2314)、続いて判断2316と呼び出し側定義の処理を実行させる(2318)。判断2316はデフォルト処理が正常に行われると、呼び出し側定義処理2318をバイパスするように働く。フォールバック・オプションがデフォルト処理を伴う呼び出し側定義であれば、スイッチ・オペレーション2308は呼び出し側定義処理を実行させ(2320)、続いて判断2322とデフォルト処理をを実行させる(2324)。判断2322は、呼び出し側定義処理2320が正常に行われたときは、デフォルト処理2324をバイパスするように働く。スイッチ・オペレーション2308に関連する処理に続いて、フォールバック処理2300がマッピングまたは変換コードを正常に特定していたかどうかの判断2326が行われる。フォールバック処理2300が失敗していれば、文字セットのデフォルト・フォールバック文字シーケンスが取得される(2328)。デフォルト・

フォールバック文字シーケンスとは、フォールバック・ルックアップ2302が変換コードを特定するのに失敗したとき使用される変換コードである。好ましくは、デフォルト・フォールバック文字シーケンスはマッピング・テーブル414のヘッダに収められている。例えば、ASCIIでは、デフォルト・フォールバック文字シーケンスは“?”であるのが普通である。そのあと、ブロック2328に続いて、またはフォールバック処理がマッピングまたは変換コードを取得するのに成功したときはブロック2326に続いて、フォールバック・オプションが使用されたことを示すエラー・コードがセットされ(2306)、フォールバック・ハンドリング処理2300は完了し、戻る。

図28はデフォルト処理2400を示すフローチャートである。デフォルト処理は図27のブロック2310、2314および2324で実行される処理と関連している。

デフォルト処理2400は最初に現在のカウンタをゼロにセットする(2402)。次に、現在のカウンタがテキスト要素の長さより大であるか等しいかの判断2404が行われる。そうであれば、デフォルト処理2400は完了し、戻る。そうでなければ、フォールバック・フラグがセットされた単一ユニコード文字に対してルックアップ処理が実行される(2406)。ここでは、ルックアップはテキスト要素の個別文字に対するものであるが、以前(ブロック3202)では、ルックアップはテキスト要素全体に対するものであった。そのあと、単一ユニコード文字の変換コードが見つかったかどうかの判断2408が行われる。見つからなければ、ユニコード文字で使用している個別マッピングがなかったことをエラー・コードで通知してデフォルト処理2400は戻る(2410)。他方、変換コードが見つかっていれば、現在のカウンタがインクリメントされ(2412)、処理はテキスト要素内の次のユニコード文字の処理を行うためにブロック2404に戻る。

上述した図4～図28はユニコードからターゲット・コード化への変換(ユニコードから)に関するものであるが、上述したように、ユニコード・コード変換システム300は異なるソース・コード化からユニコードに変換する(ユニコードへ)機能も等しく備えている。ユニコードへはユニコードからの処理と類似し

ているが、実質的には複雑度が軽減されている。ユニコードへの処理はターゲット・コード化を判断するときテキスト要素を探すためにスキャンしたり、複数の文字シーケンスを調べたりする必要がないのが通常である。ユニコードへの処理はソース文字列を個々の文字に分割し、そのあとでユニコードの対応するコード・ポイントを見つけるだけである。しかし、文字のマッピングがそのあとに続く文字に影響されるようなまれな場合には（例えば、デーヴァナーガリーのようなインド・スクリプト）、上述したようにスキャンが行われる場合がある。

図29は本発明によるユニコード・コード変換システム2500の実施例を示すブロック図である。ユニコード・コード変換システム2500はユニコードへの変換を行う（つまり、ユニコードへの処理）。ユニコード・コード変換システム2500はユニコードへのコンバータ2502を含み、このコンバータはソース文字列2504を受信し、ユニコード文字列2506を出力する。ユニコードへのコンバータ2502はスキャナ2508とやりとりするユニコードへのコンバータを通してコード変換プロセスを実行する。スキャナ2508はスキャナ・テーブル2510を使用して、ソース文字列2504をスキャンしソース文字列2504を文字に分断する。ここでは、ユニコードからの場合と異なり、ソース文字列は個々の文字に分割されるだけである。次に、ユニコードへのコンバータ2502はルックアップ・ハンドラ2512を使用して個別文字を調べそのユニコード・コード化を取得する。ルックアップ・ハンドラ2512はマッピング・テーブル2514を使用してユニコードの文字を取得する。さらに、ユニコードへのコンバータ2502はフォールバック・ハンドラ2516を使用することもできる。フォールバック・ハンドラ2516はマッピング・テーブル2514と一緒に動作して、ルックアップ・ハンドラ2512がユニコード文字を特定できなかった場合に、テキスト要素のフォールバック・マッピングとして使用できるターゲット・コード化内の1つまたは複数の文字を特定する。

スキャナ2508、スキャナ・テーブル2510、ルックアップ・テーブル2512、マッピング・テーブル2514、フォールバック・ハンドラ2516、状態管理機構2518は図4の対応するデバイスと類似しているが、実質的には複雑度が軽減されている。従って、これらのデバイスはユニコードへ

とユニコードからの両方の処理を実現するように設計することができる。さらに、与えられたコンピュータ・システムや他のエレクトロニック・デバイスがユニコードへとユニコードからの両方の処理を実行する機能を備えているときは、そのコンピュータ・システムや他のエレクトロニック・デバイスはハブとして動作することができる。このハブはユニコードがサポートする種々の各国語文字間で変換を行うように動作することができる。例えば、ソースの各国語文字セットはまずユニコードに変換され、次にターゲットの各国語文字セットに変換される。

本発明の第1側面によれば、上述したテーブルは、好ましくは、上述したフォーマットを使用してテーブルにストアすべきデータの圧縮を行う。さらに、本発明の第1側面によれば、圧縮を重視している。例えば、圧縮すると、属性テーブルは50倍(50対1)に小さくなる。しかし、本発明の第1側面によれば、この分野の精通者ならば理解されるように、テーブルのフォーマットは多数の形体をとることができるので、あるテーブルは他のテーブルよりも実現が容易になる。最も容易な実現はどの圧縮も使用しないテーブルであるが、そのためには最大のデータ記憶装置が必要になる。さらに、本発明の第1側面によれば、テーブルは上述したインプリメンテーションで使用されているが、これらのテーブルで引き起こされる振舞を直接にコーディングすることも可能である。しかし、直接コーディングはコード変換システムのオペレーションの変更を困難にするのに対し、テーブルを使用すると、そのような変更を取り入れるためにテーブルだけを変更するだけで済むのが普通である。

本発明の多数の特徴および利点は上述してきた説明で明らかにした通りであり、本発明のかかる特徴と利点はすべて請求の範囲に記載されている。さらに、本発明はこの分野の精通者が容易に理解されるように、種々態様に変更することが可能であるので、本発明は上述してきた正確な構造とオペレーションに限定されるものではない。従って、すべての適当な変更および等価技術は本発明の範囲に属するものである。

【図1】

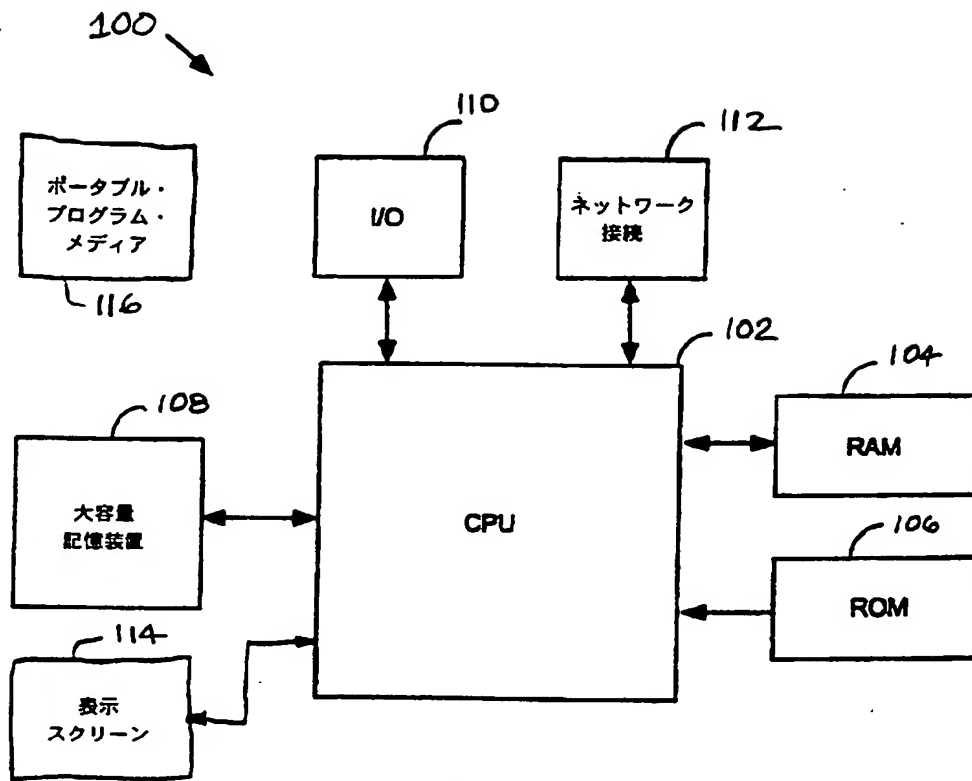


Figure 1

【図2】

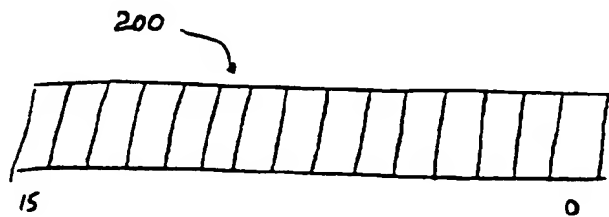


FIG. 2

【図3】

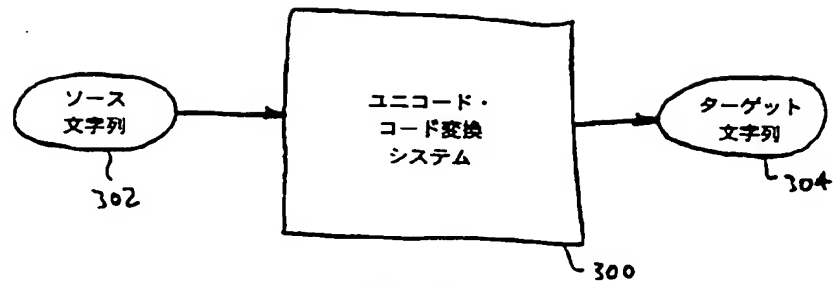


FIG. 3

【図4】

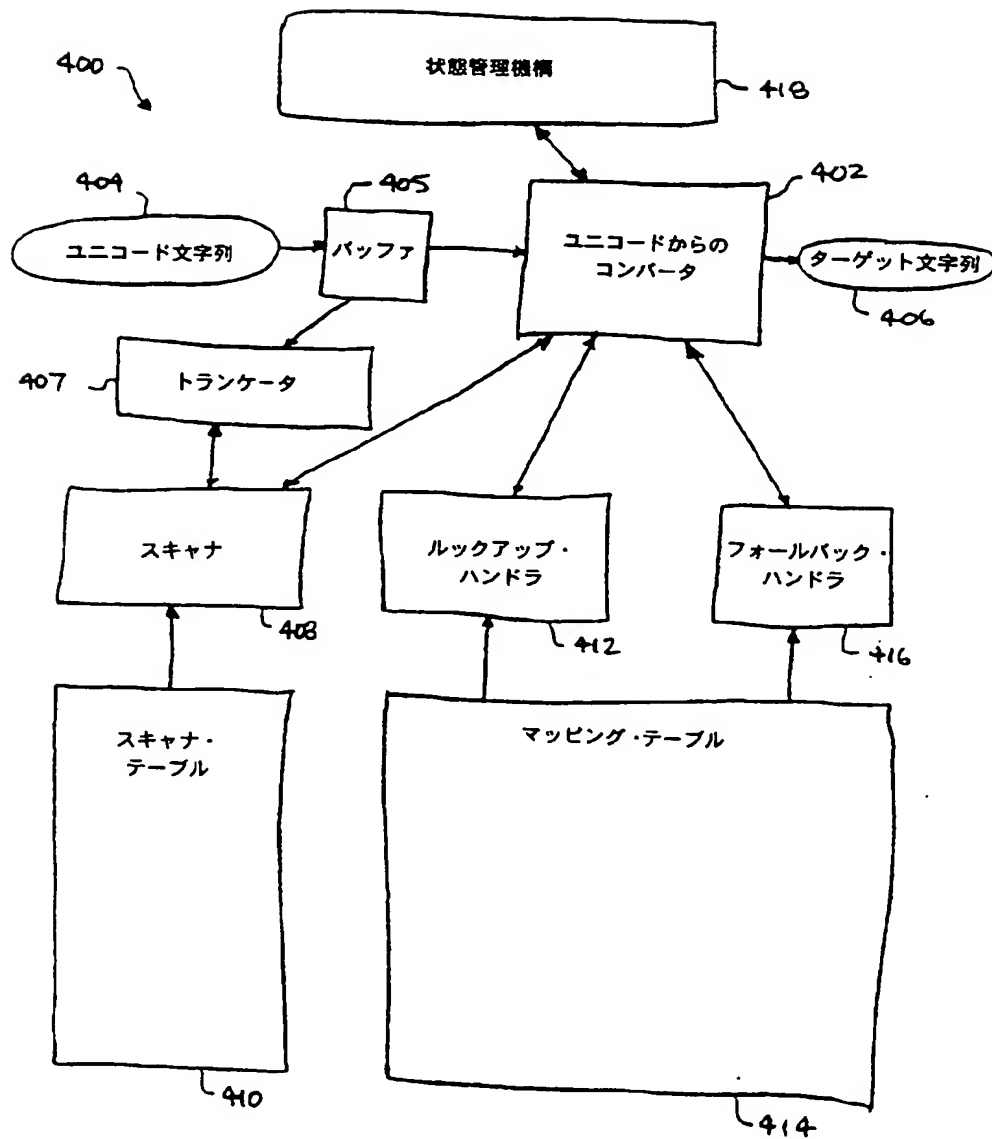


FIG. 4

【図5】

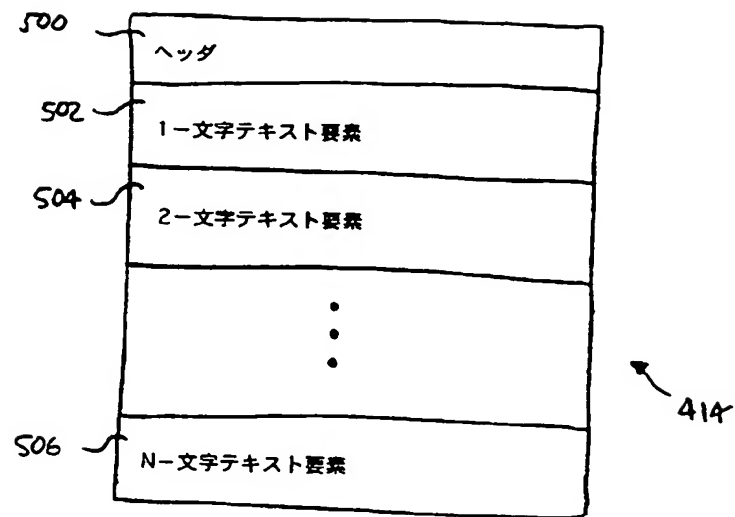
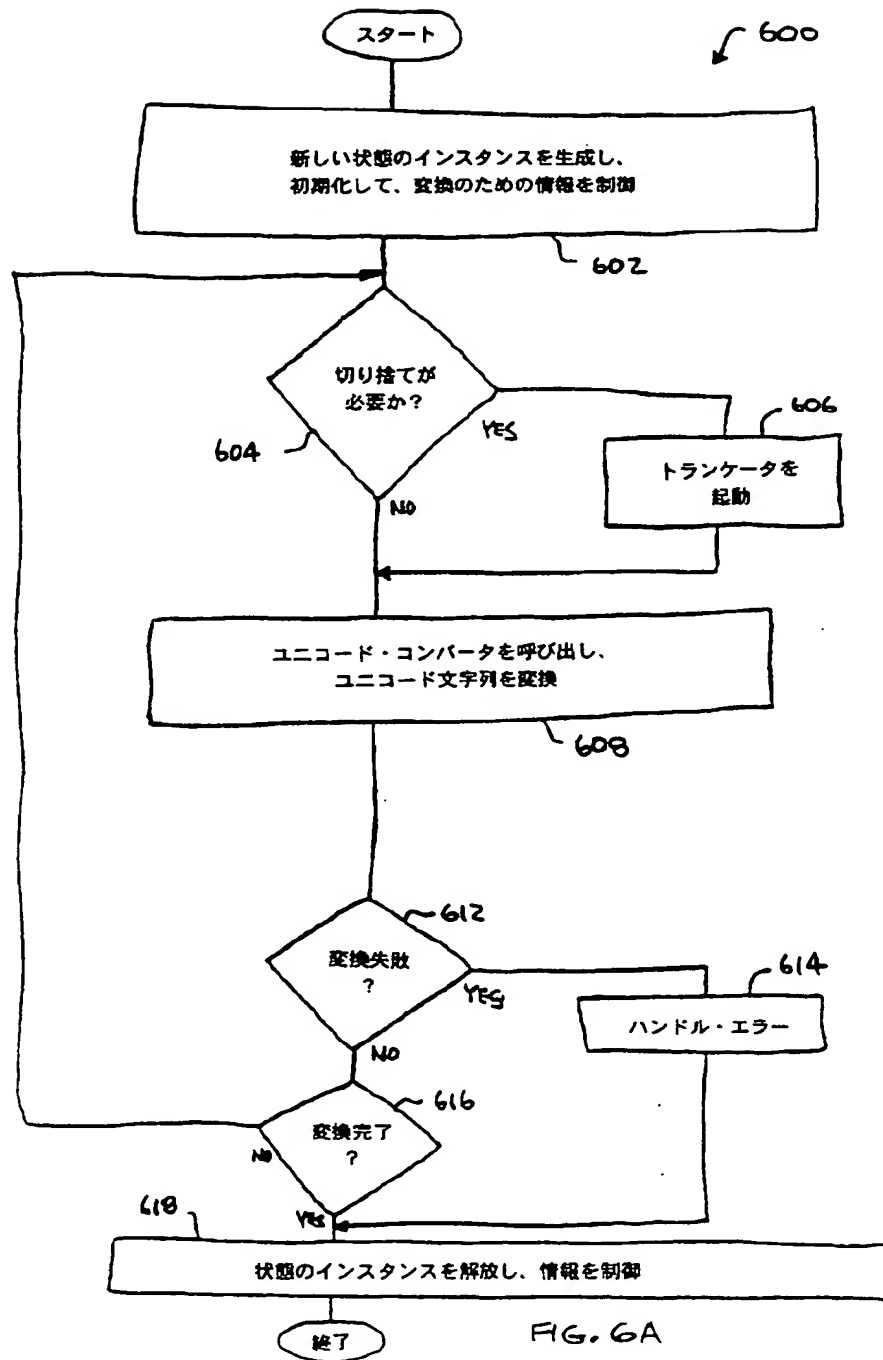


FIG. 5

【図6】



【図6】

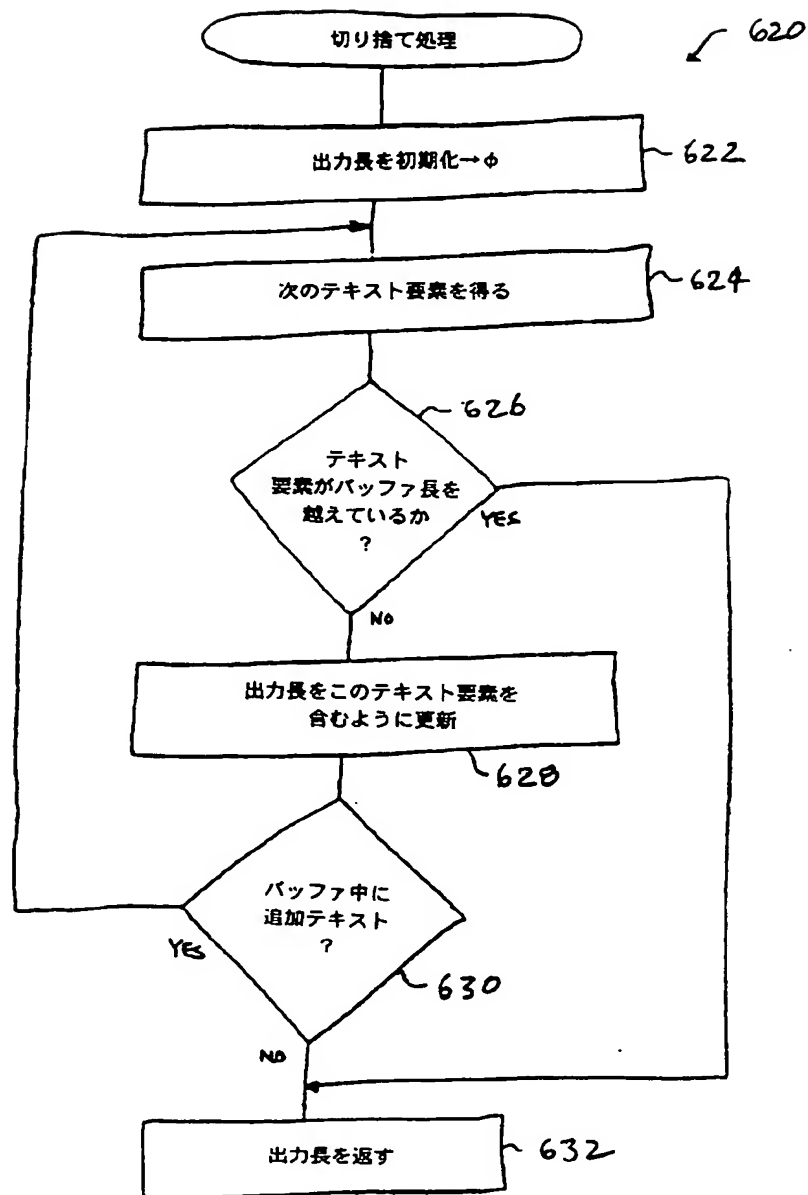


FIG. 6B

【図7】

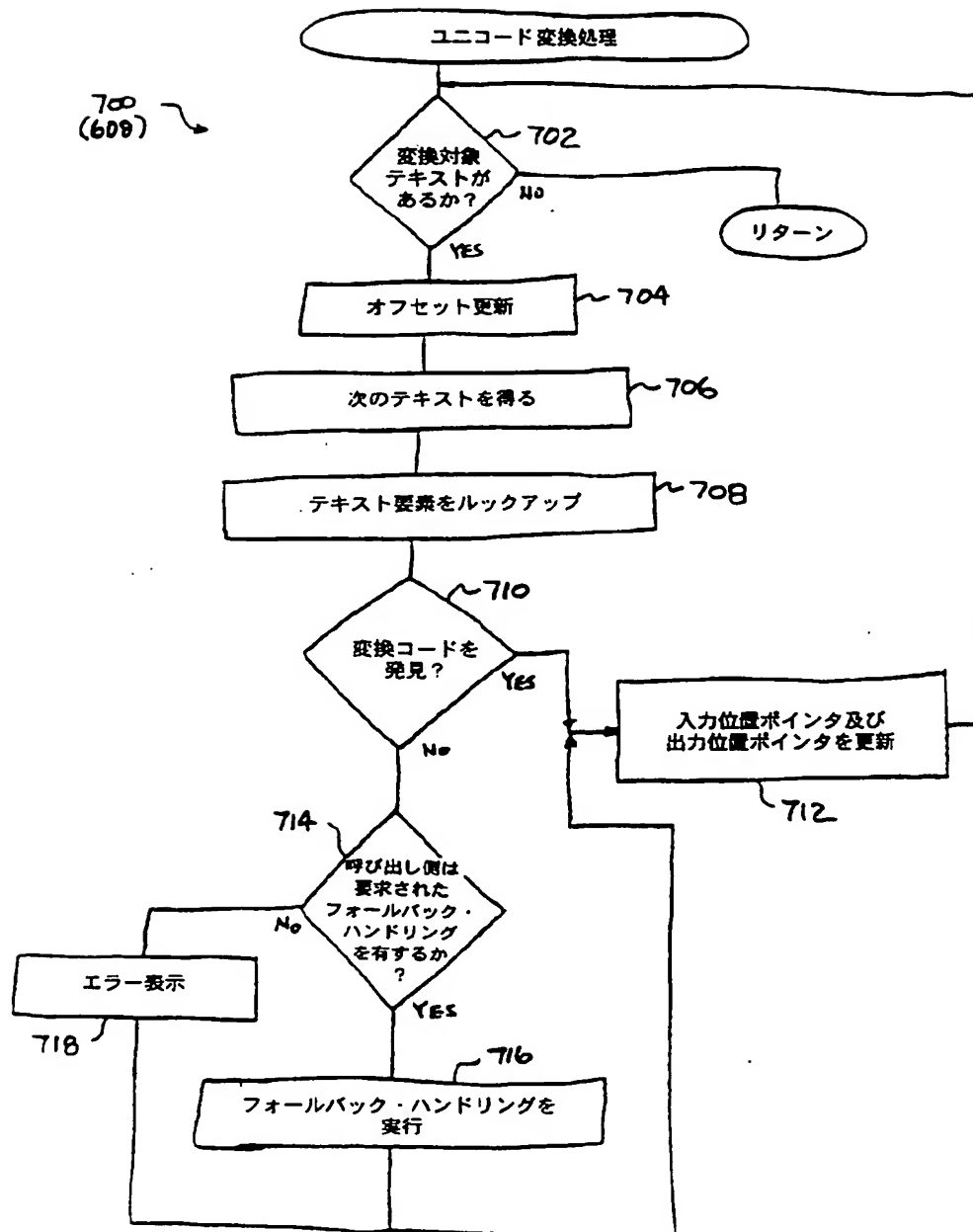


FIG. 7

【図8】

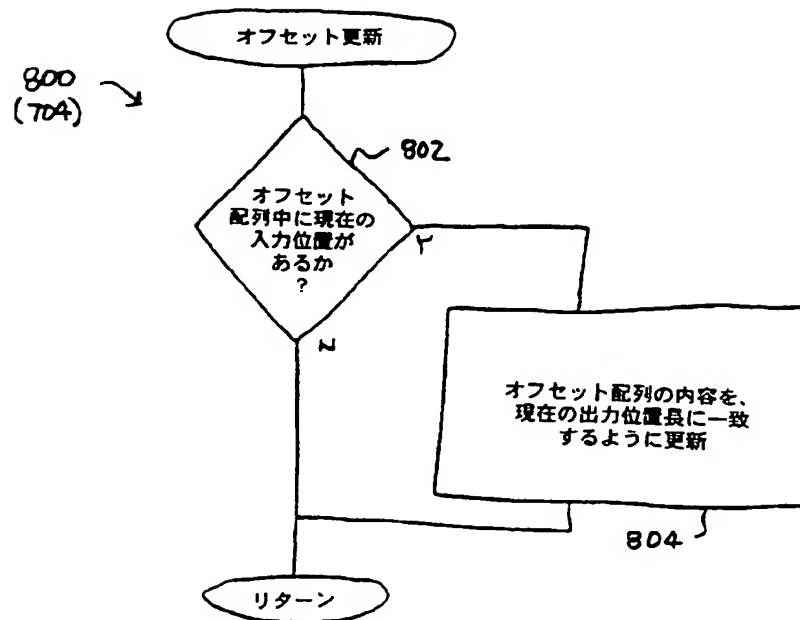


FIG. 8

【図9】

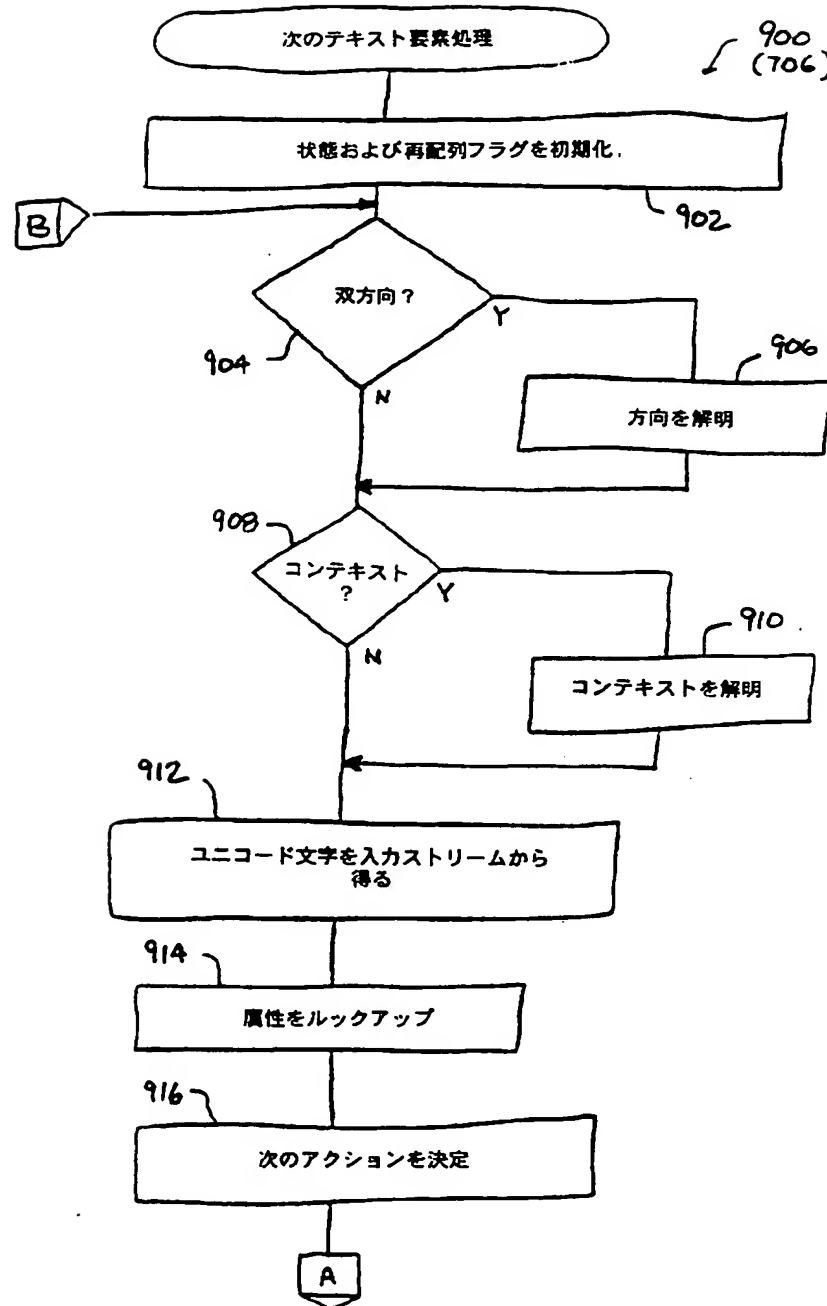


FIG. 9A

【図9】

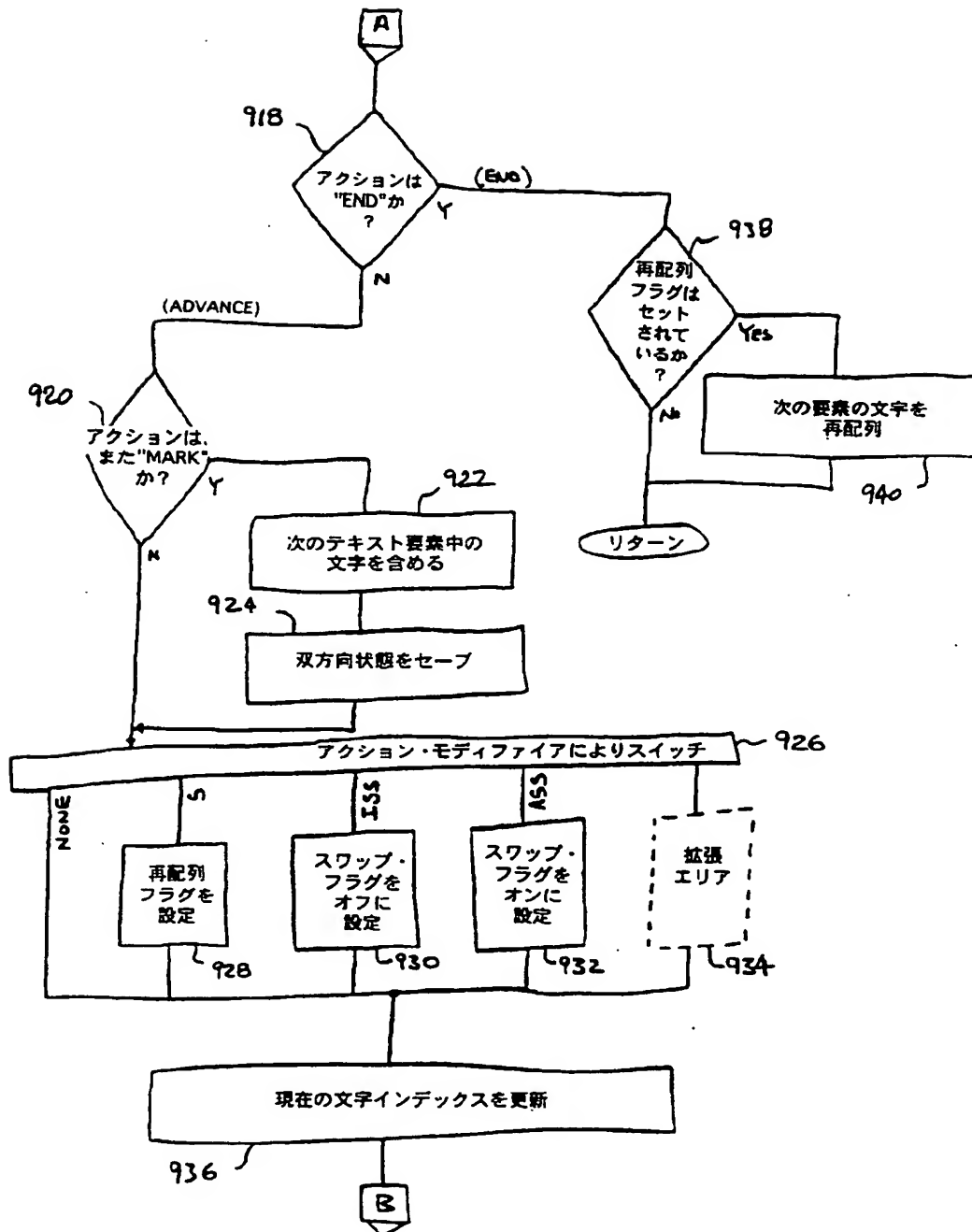
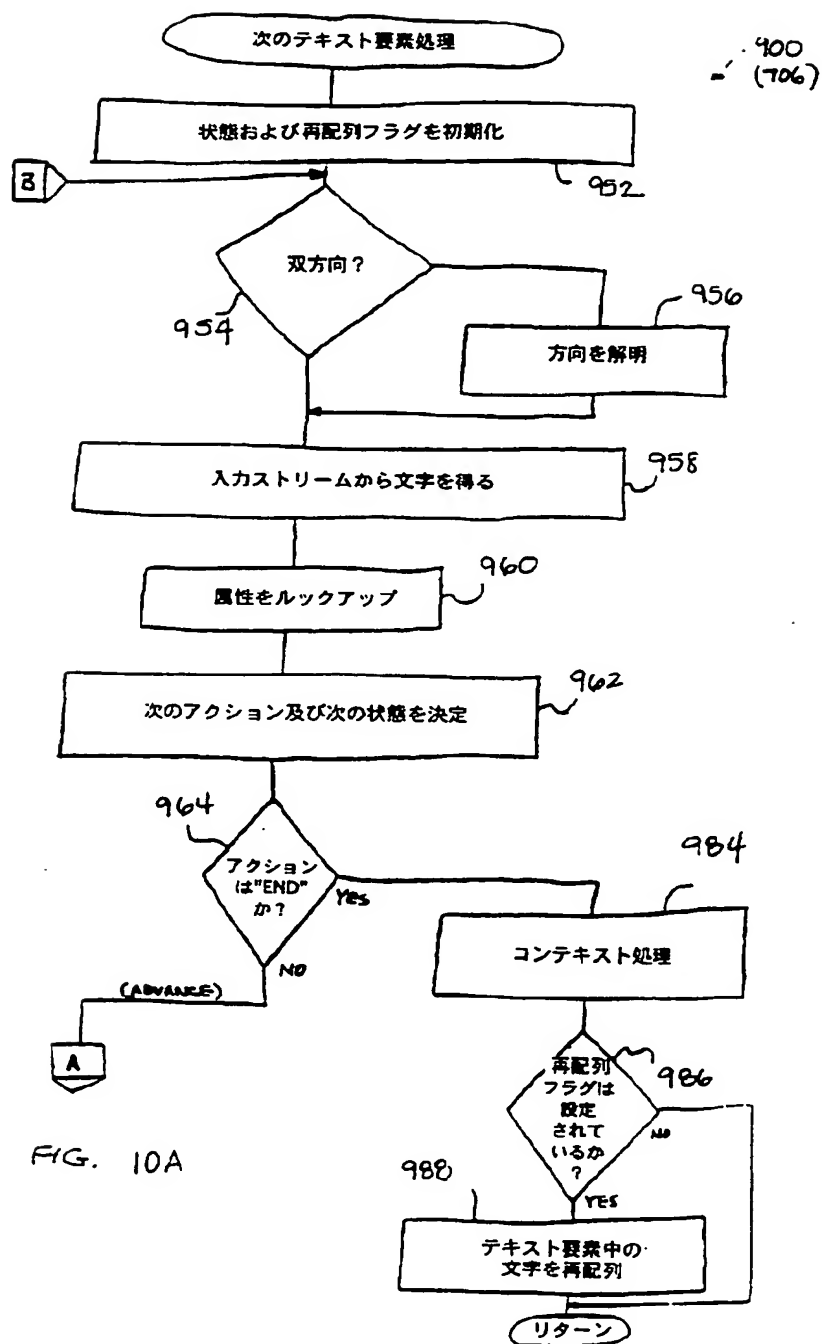


FIG. 9B

【図10】



【図10】

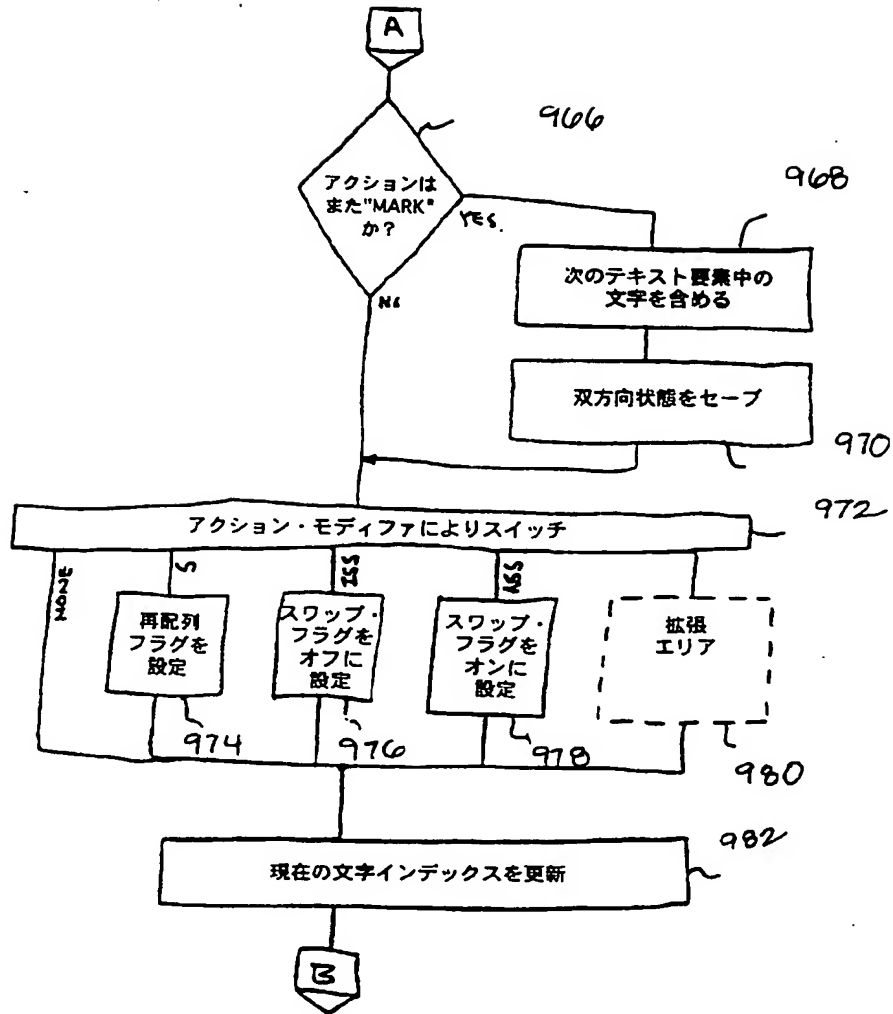
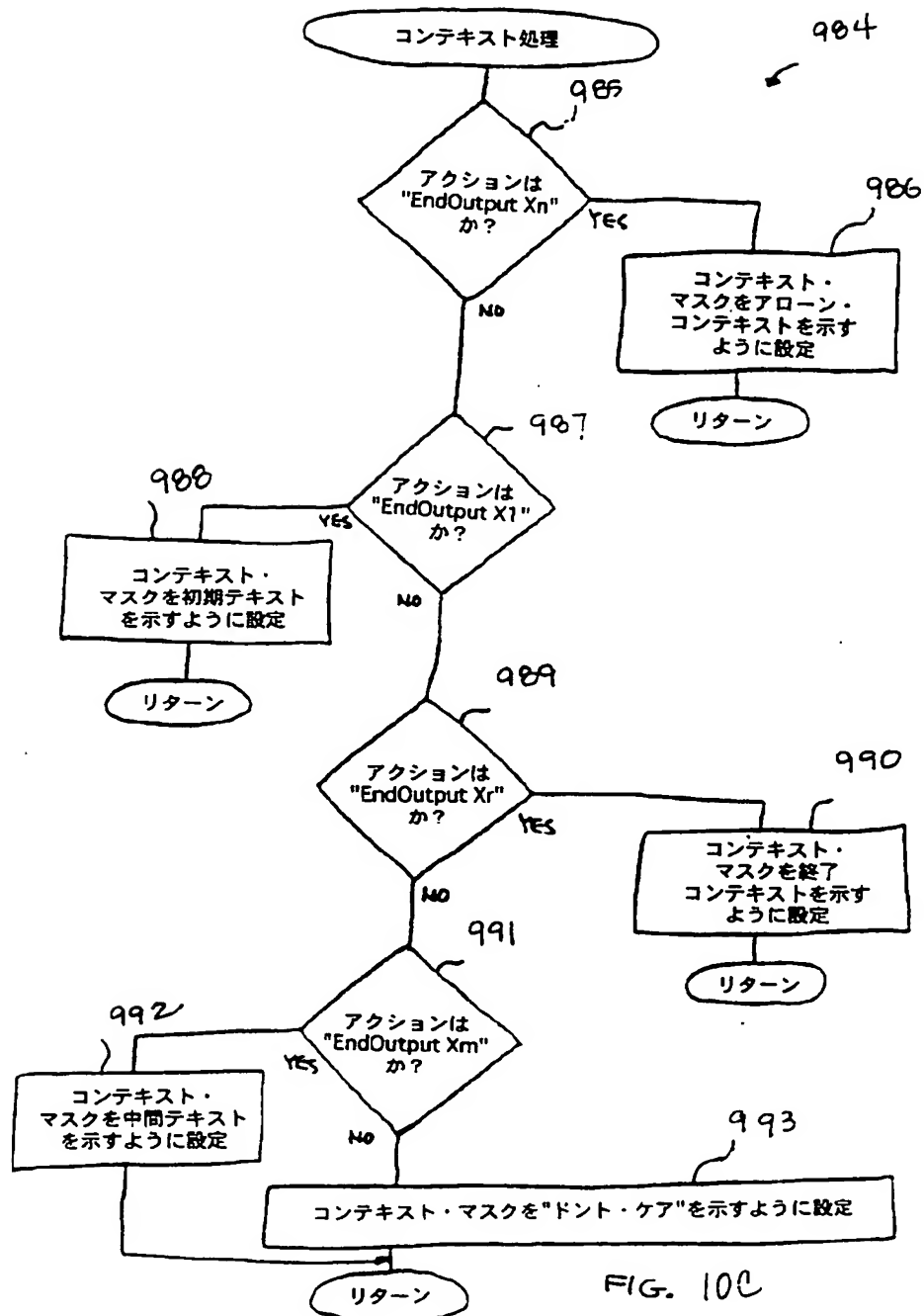


FIG. 10B

【図10】



【図11】

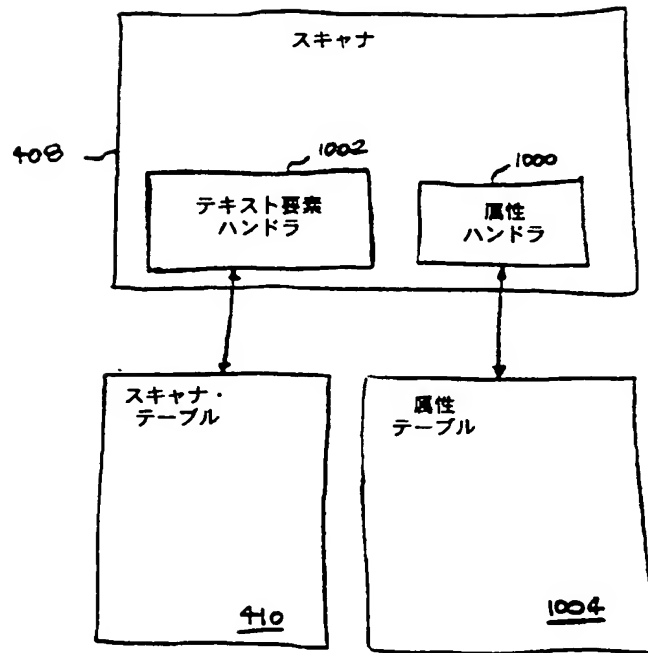


FIG. 11

【図12】

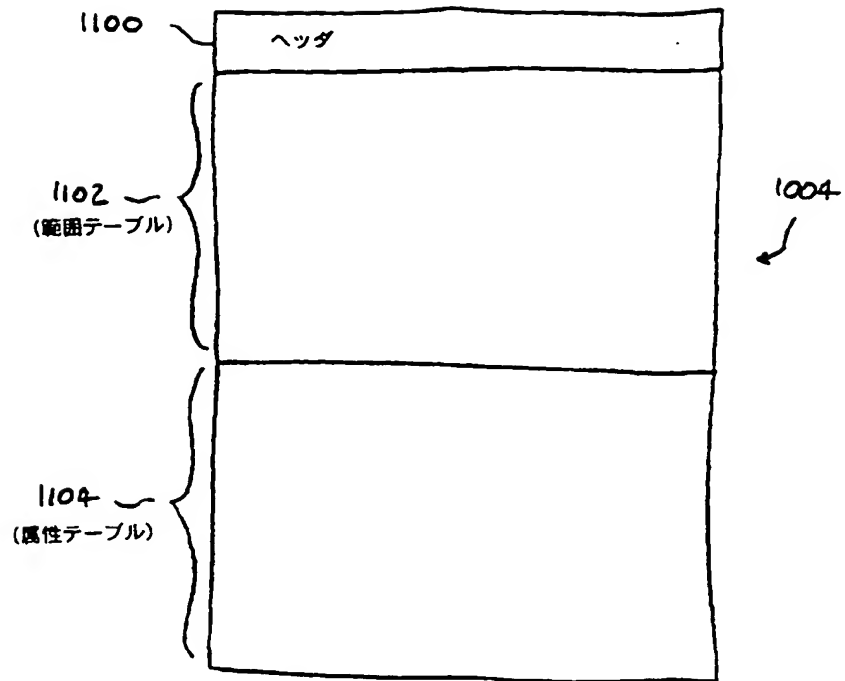


FIG. 12

【図13】

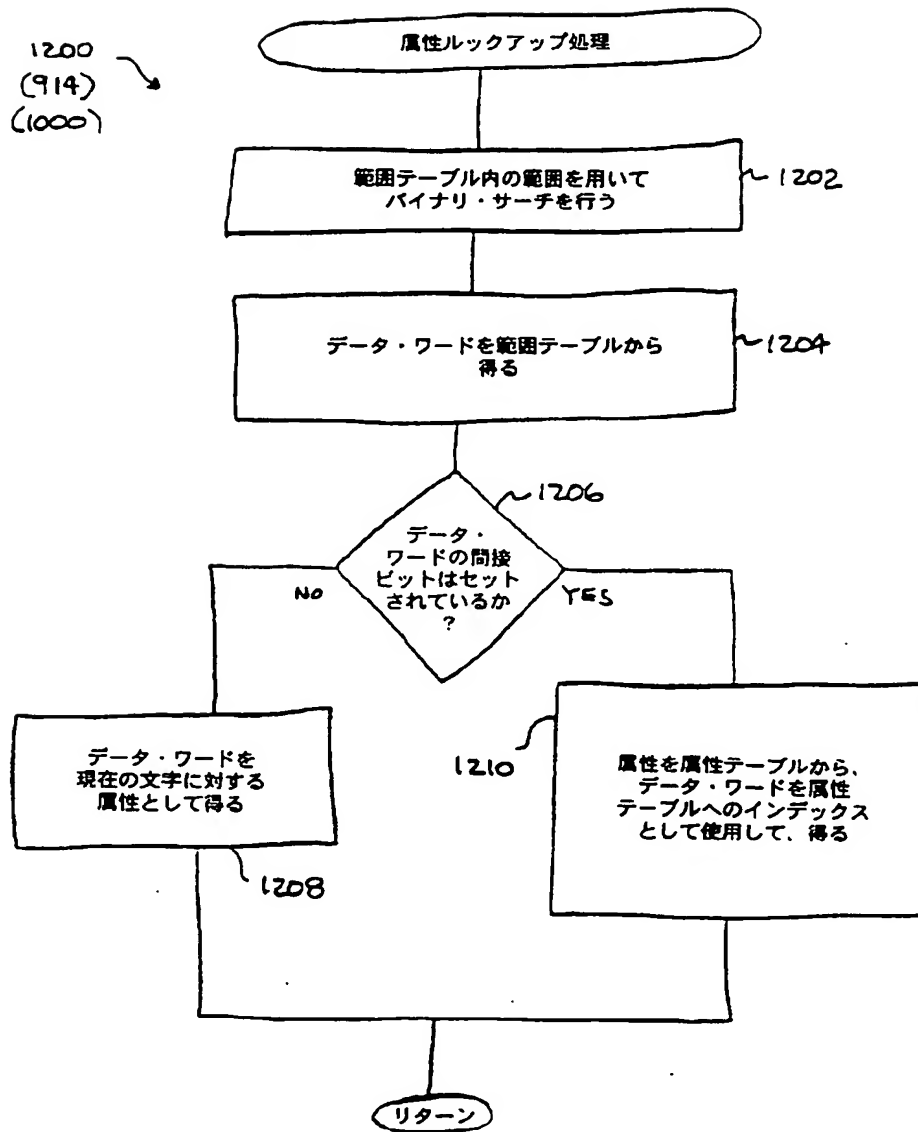


FIG. 13

【図14】

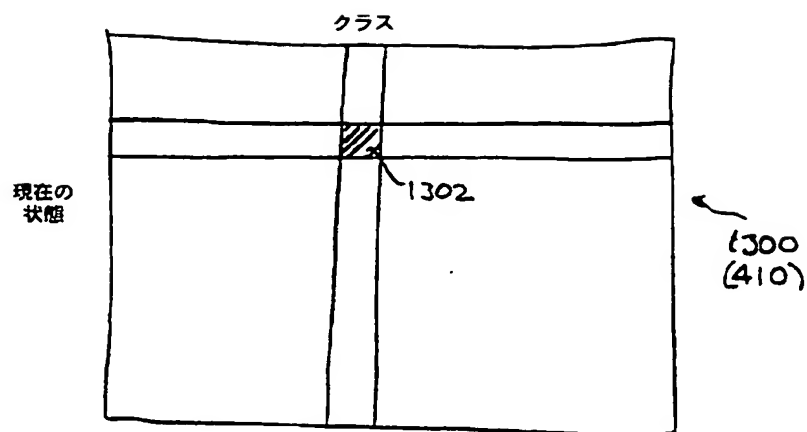


FIG. 14A

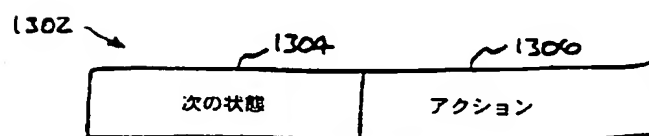


FIG. 14B

【 図 1 5 】

1400 →

状態	OS	N3	DD	HD	LD	FT	FS	CC	ISS	ASS	NU
0=End	End	End	End	End	End	End	End	End	End	End	End
	->0	->0	->0	->0	->0	->0	->0	->0	->0	->0	->0
1=Start	AdvMark	AdvMark	AdvMark	AdvMark	AdvMark	AdvMark	AdvMark	AdvMark	AdvMark	AdvMark	AdvMark
	->2	->2	->2	->2	->2	->2	->2	->2	->2	->2	->2
1=TEMS	End	AdvMark	AdvMark	AdvMark	End	End	End	End	End	End	End
	->0	->2	->2	->2	->0	->0	->0	->0	->0	->0	->0
3=LD	End	AdvMark	AdvMark	AdvMark	End	End	Adv	End	End	End	End
	->0	->2	->2	->2	->0	->0	->0	->0	->0	->0	->0
6=LD,LD+	End	End	End	End	End	End	Adv	End	End	End	End
	->0	->0	->0	->0	->0	->0	->0	->0	->0	->0	->0
5=LD,FS	End	End	End	End	End	End	Adv	End	End	End	End
	->0	->0	->0	->0	->0	->0	->0	->0	->0	->0	->0
6=LD,FS,LD+	End	AdvMark	AdvMark	AdvMark	End	End	AdvMark	End	End	End	End
	->0	->2	->2	->2	->0	->0	->0	->0	->0	->0	->0
7=IL	End	AdvMark	AdvMark	AdvMark	AdvMark	End	End	End	End	End	End
	->0	->2	->2	->2	->2	->0	->0	->0	->0	->0	->0
8=IL,IV	End	AdvMark	AdvMark	AdvMark	AdvMark	AdvMark	End	End	End	End	End
	->0	->2	->2	->2	->2	->2	->0	->0	->0	->0	->0
9=IL,IV,IT	End	AdvMark	AdvMark	AdvMark	AdvMark	AdvMark	End	End	End	End	End
	->0	->2	->2	->2	->2	->2	->0	->0	->0	->0	->0

FIG. 15

【図 16】

		SUMMER TIME OCCURRING CHANGES																															
		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Address	10000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10001	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10002	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10003	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10004	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10005	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10006	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10007	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10008	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10009	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10010	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10011	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10012	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10013	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10014	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10015	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10016	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10017	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10018	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10019	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10020	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10021	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10022	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10023	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10024	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10025	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10026	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10027	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10028	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10029	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10030	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10031	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10032	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10033	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10034	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10035	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10036	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10037	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10038	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10039	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10040	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10041	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
Address	10042	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000													

FIG. 16A

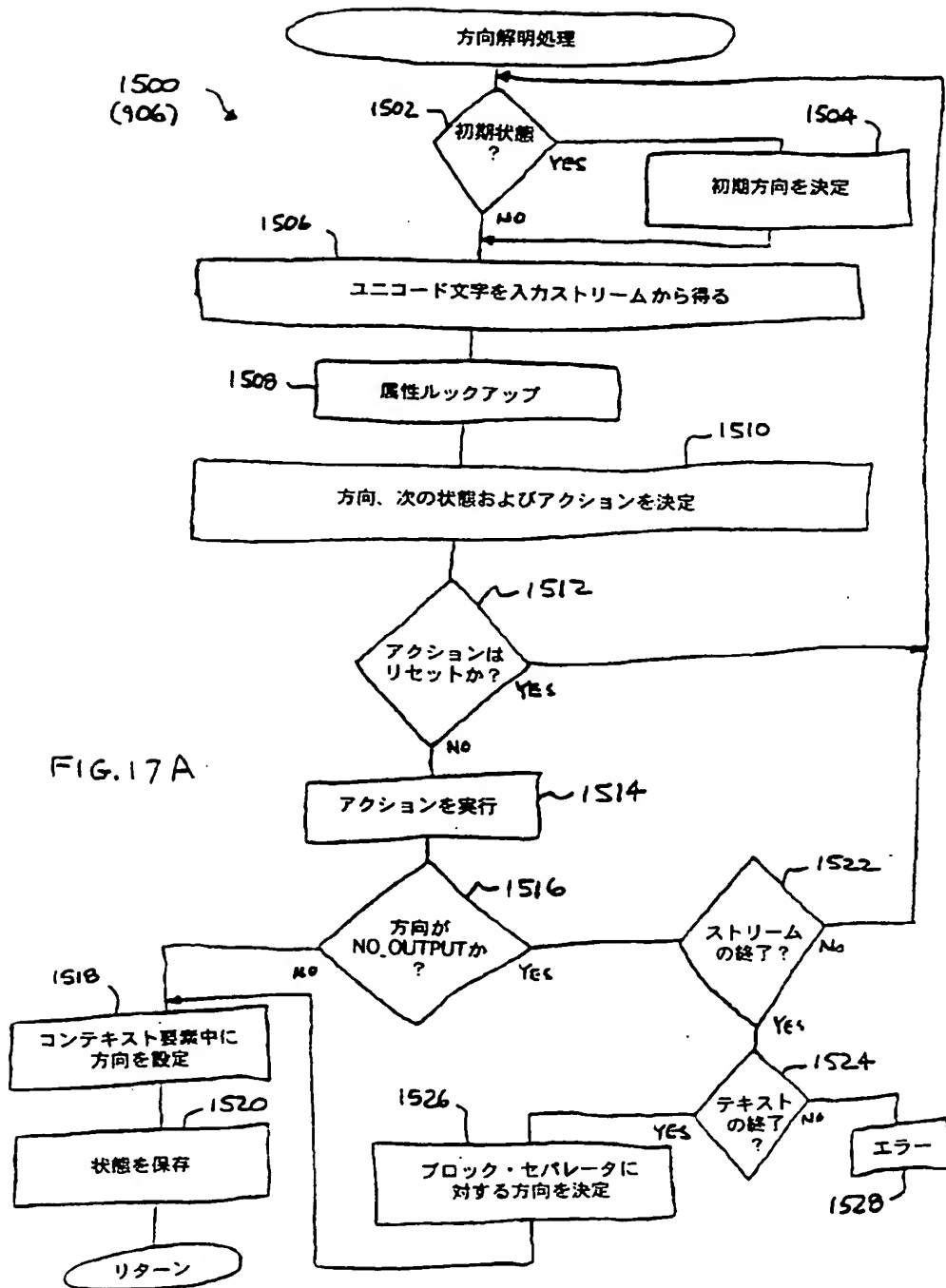
16A 16B

【图 16】

[illegible]

FIG. 16B

【図17】



【 図 1 7 】

	START	sDIR	OR	SL	SR	AL	L+AN	L+EN
LR	SL,L	START,L	ORG	SL,L	SL,L	SL,L	SL,L	SL,L
RL	SR,R	START,R	ORG	SR,R	SR,R	SR,R	SR,R	SR,R
AL	AL,L	START,R	ORG	AL,R	AL,R	AL,R	AL,R	AL,R
LRE	SL,push,L	START,L	SL,push,L	SL,push,L	SL,push,L	SL,push,L	SL,push,L	SL,push,L
RLE	SR,push,R	START,R	SR,push,R	SR,push,R	SR,push,R	SR,push,R	SR,push,R	SR,push,R
LRO	OR,push,L	START,L	OR,push,L	OR,push,L	OR,push,L	OR,push,L	OR,push,L	OR,push,L
RLO	OR,push,R	START,R	OR,push,R	OR,push,R	OR,push,R	OR,push,R	OR,push,R	OR,push,R
PDF	S(G),pop,G	error	S(G),pop,G	S(G),pop,G	S(G),pop,G	S(G),pop,G	S(G),pop,G	S(G),pop,G
AN	(G)+AN,L	sDIR,*	ORG	L+AN,L	R+AN,L	AL+Num,L	L+AN,L	L+AN,L
EN	(G)+EN,L	sDIR,*	ORG	L+EN,L	R+EN,L	AL+Num,L	L+EN,L	L+EN,L
ET	s#+ET,*	sDIR,*	ORG	sl+N,*	sr+ET,*	sa+N,*	slAN+T,*	L+EN,L
ES	s(G)+N,*	sDIR,*	ORG	sl+N,*	sr+N,*	sa+N,*	slAN+N,*	slEN+N,*
CS	s(G)+N,*	sDIR,*	ORG	sl+N,*	sr+N,*	sa+N,*	slAN+S,*	slEN+N,*
ON	s(G)+N,*	sDIR,*	ORG	sl+N,*	sr+N,*	sa+N,*	slAN+N,*	slEN+N,*
BS	S(G),G	sDIR,*	reset	reset	reset	reset	reset	reset
Notes:		scan for base	override	strong left	strong right	arabic letter	strong left+ arabic num	strong left+ euro num
		direction						

FIG. 17B

1511

17B	17C	17D
-----	-----	-----

R+AN	R+EN	AL+Num	SL+N	SR+N	SR+ET	SA+N	SR+ET	SLAN+N
SL,L	SL,L	SL,L	SL,L	SR,G	SR,G	AL,G	S(G),G	L+AN,G
SR,R	SR,R	SR,R	SL,G	SR,R	SR,R	AL,R	S(G),G	L+AN,R
AL,R	AL,R	AL,R	SL,G	SR,R	SR,R	AL,R	S(G),G	L+AN,R
SL,push,L	SL,push,L	SL,push,L	SL,L	SR,G	SR,G	AL,G	S(G),G	L+AN,G
SR,push,R	SR,push,R	SR,push,R	SL,G	SR,R	SR,R	AL,R	S(G),G	L+AN,R
OR,push,L	OR,push,L	OR,push,L	SL,L	SR,G	SR,G	AL,G	S(G),G	L+AN,G
OR,push,R	OR,push,R	OR,push,R	SL,G	SR,R	SR,R	AL,R	S(G),G	L+AN,R
S(G),pop,G	S(G),pop,G	S(G),pop,G	SL,G	SR,G	SR,G	AL,G	S(G),G	L+AN,G
R+AN,L	R+AN,L	AL+Num,L	SL,G	SR,R	SR,R	AL,R	S(G),G	L+AN,R
R+EN,L	R+EN,L	AL+Num,L	SL,L	SR,G	SR,L	AL,R	S(G),L	L+AN,G
SRAN+T,•	R+EN,L	sALN+N,•	SL+N,•	SR+N,•	SR+ET,•	SA+N,•	SR+ET,•	SLAN+N,•
SRAN+N,•	sREN+S,•	sALN+N,•	SL+N,•	SR+N,•	SR+N,•	SA+N,•	S(G)+N,•	SLAN+N,•
SRAN+N,•	sREN+S,•	sALN+S,•	SL+N,•	SR+N,•	SR+N,•	SA+N,•	S(G)+N,•	SLAN+N,•
SRAN+N,•	sREN+N,•	sALN+N,•	SL+N,•	SR+N,•	SR+N,•	SA+N,•	S(G)+N,•	SLAN+N,•
reset	reset	reset	SL,G	SR,G	SR,G	AL,G	S(G),G	L+AN,G
strong right+ arabic num	strong right+ euro num	arabic letter+ any num	lockheads for random	neutrals	bot+euro term very special			

1511

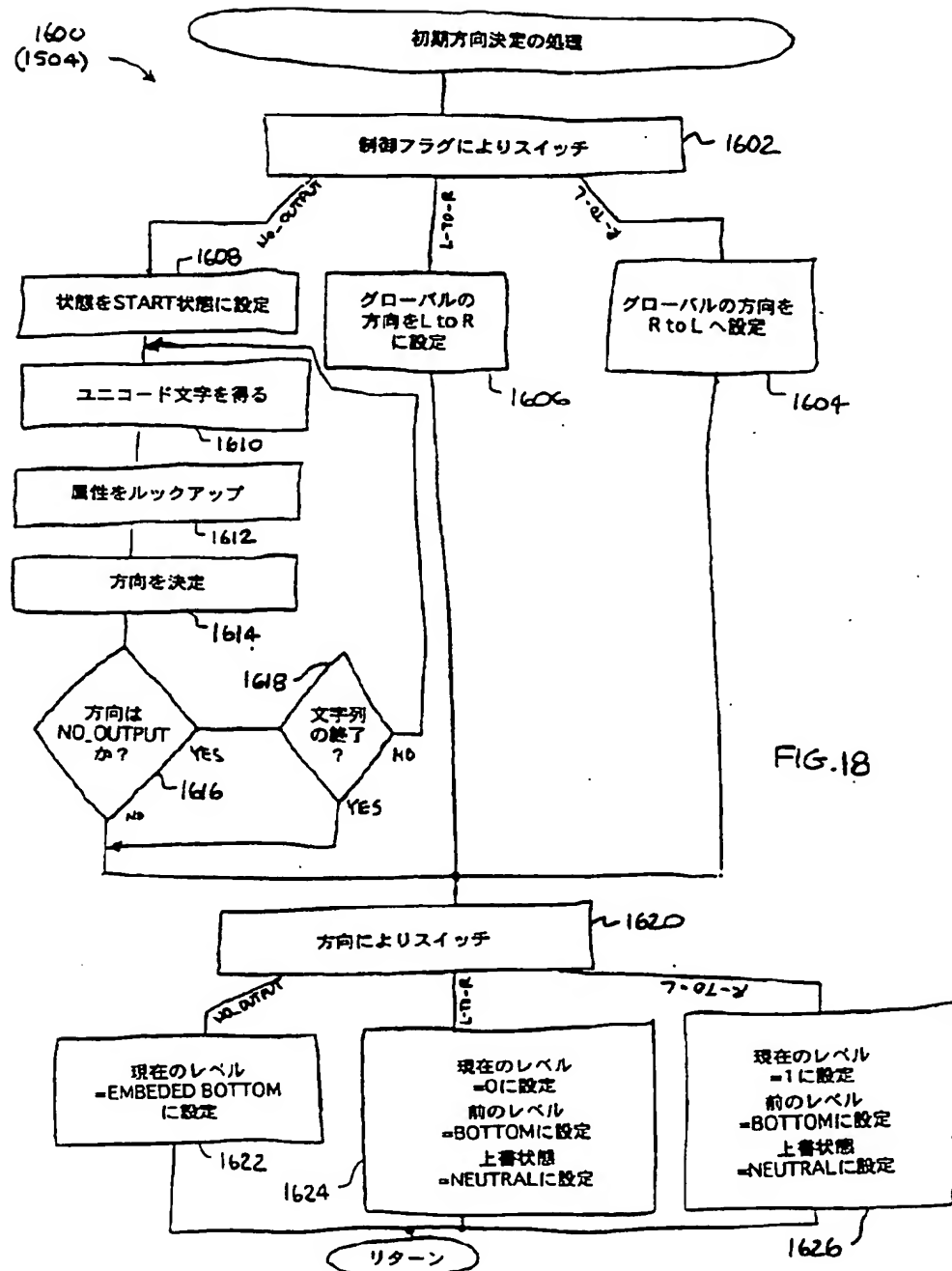
FIG. 17C

1511

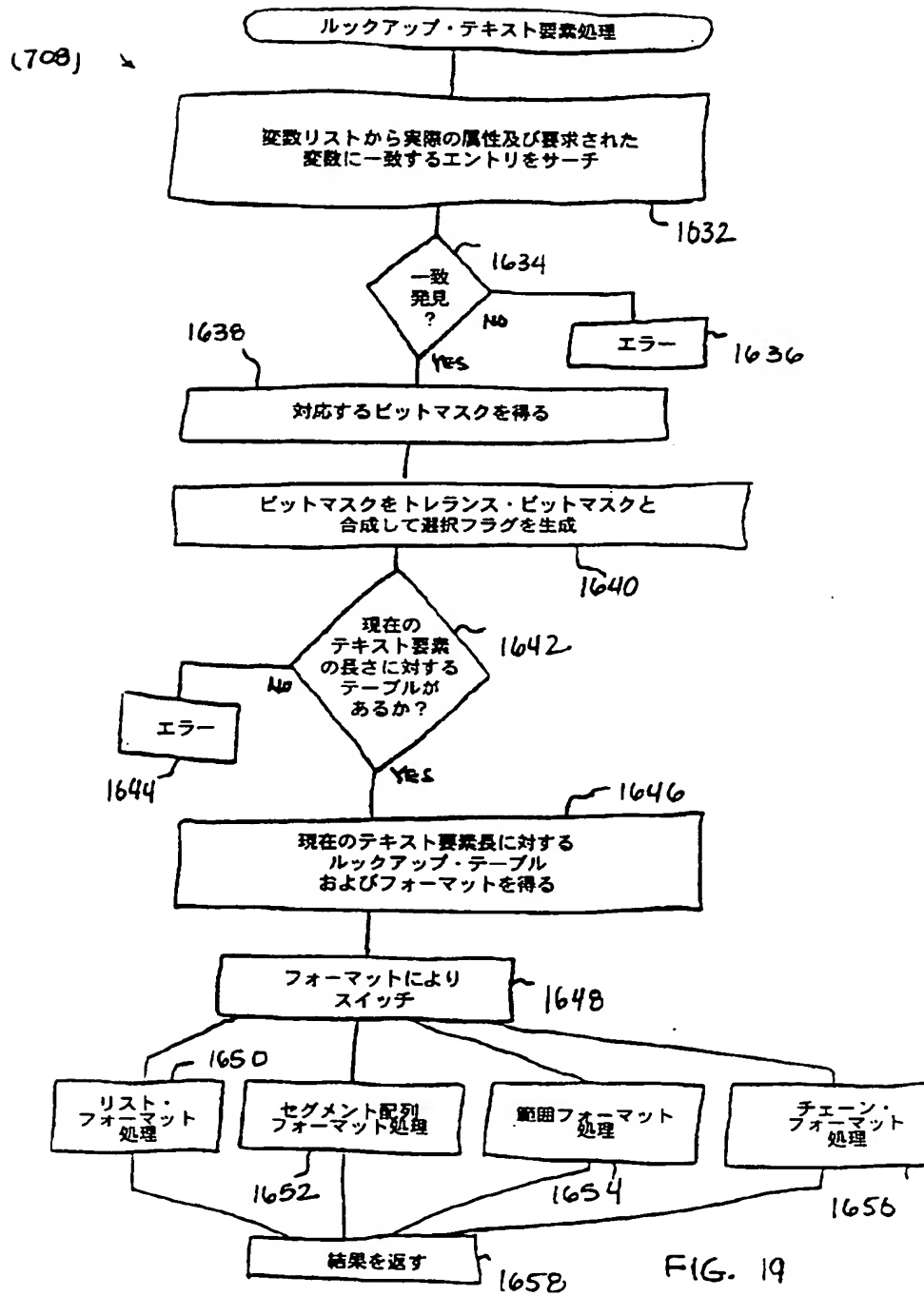
sLEN+N	sRAN+N	sREN+N	sALN+N	sLAN+T	sRAN+T	sLAN+S	sREN+S	sALN+S
L+EN,L	R+AN,G	R+ENG	AL+Num,G	L+AN,G	R+AN,G	L+AN,G	R+ENG	AL+Num,G
L+EN,G	R+AN,R	R+EN,R	AL+Num,R	L+AN,R	R+AN,R	L+AN,R	R+EN,R	AL+Num,R
L+EN,G	R+AN,R	R+EN,R	AL+Num,R	L+AN,R	R+AN,R	L+AN,R	R+EN,R	AL+Num,R
L+EN,L	R+AN,G	R+ENG	AL+Num,G	L+AN,G	R+AN,G	L+AN,G	R+ENG	AL+Num,G
L+EN,G	R+AN,R	R+EN,R	AL+Num,R	L+AN,R	R+AN,R	L+AN,R	R+EN,R	AL+Num,R
L+EN,L	R+AN,G	R+ENG	AL+Num,G	L+AN,G	R+AN,G	L+AN,G	R+ENG	AL+Num,G
L+EN,G	R+AN,R	R+EN,R	AL+Num,R	L+AN,R	R+AN,R	L+AN,R	R+EN,R	AL+Num,R
L+EN,G	R+AN,R	R+EN,R	AL+Num,R	L+AN,R	R+AN,R	L+AN,R	R+EN,R	AL+Num,R
L+EN,L	R+AN,R	R+EN,R	AL+Num,R	L+AN,L	R+AN,L	L+AN,G	R+EN,L	AL+Num,L
sLEN+N,•	sRAN+N,•	sREN+N,•	sALN+N,•	sLAN+T,•	sRAN+T,•	sLAN+N,•	sREN+N,•	sALN+N,•
sLEN+N,•	sRAN+N,•	sREN+N,•	sALN+N,•	sLAN+N,•	sRAN+N,•	sLAN+N,•	sREN+S,•	sALN+S,•
sLEN+N,•	sRAN+N,•	sREN+N,•	sALN+N,•	sLAN+N,•	sRAN+N,•	sLAN+S,•	sREN+S,•	sALN+S,•
sLEN+N,•	sRAN+N,•	sREN+N,•	sALN+N,•	sLAN+N,•	sRAN+N,•	sLAN+N,•	sREN+N,•	sALN+N,•
L+EN,G	R+AN,G	R+ENG	AL+Num,G	L+AN,G	R+AN,G	L+AN,G	R+ENG	AL+Num,G
				euro terms		separators		
				looks after for numeric		punctuation		

FIG. 17D

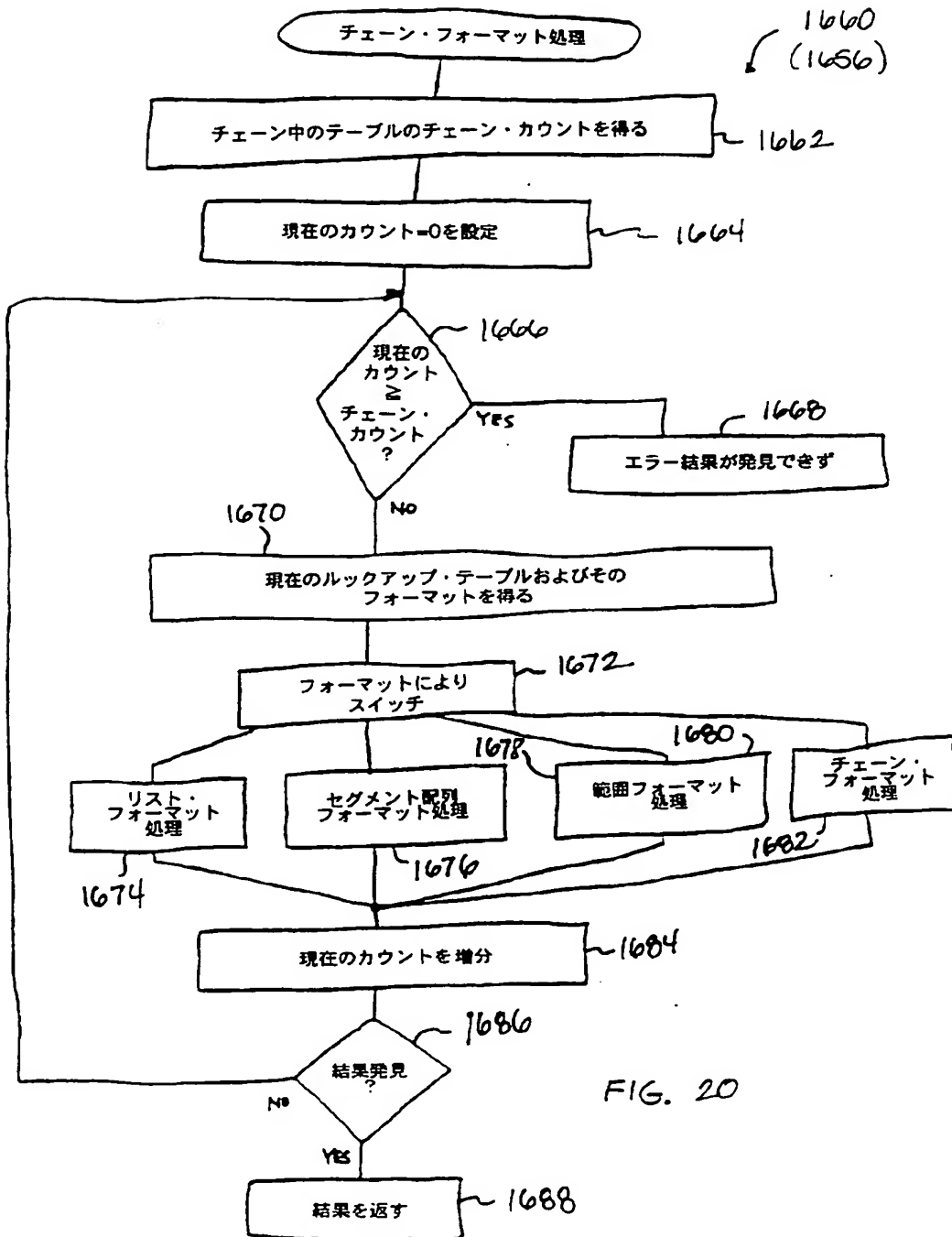
【図18】



【図19】



【図20】



【図 21】

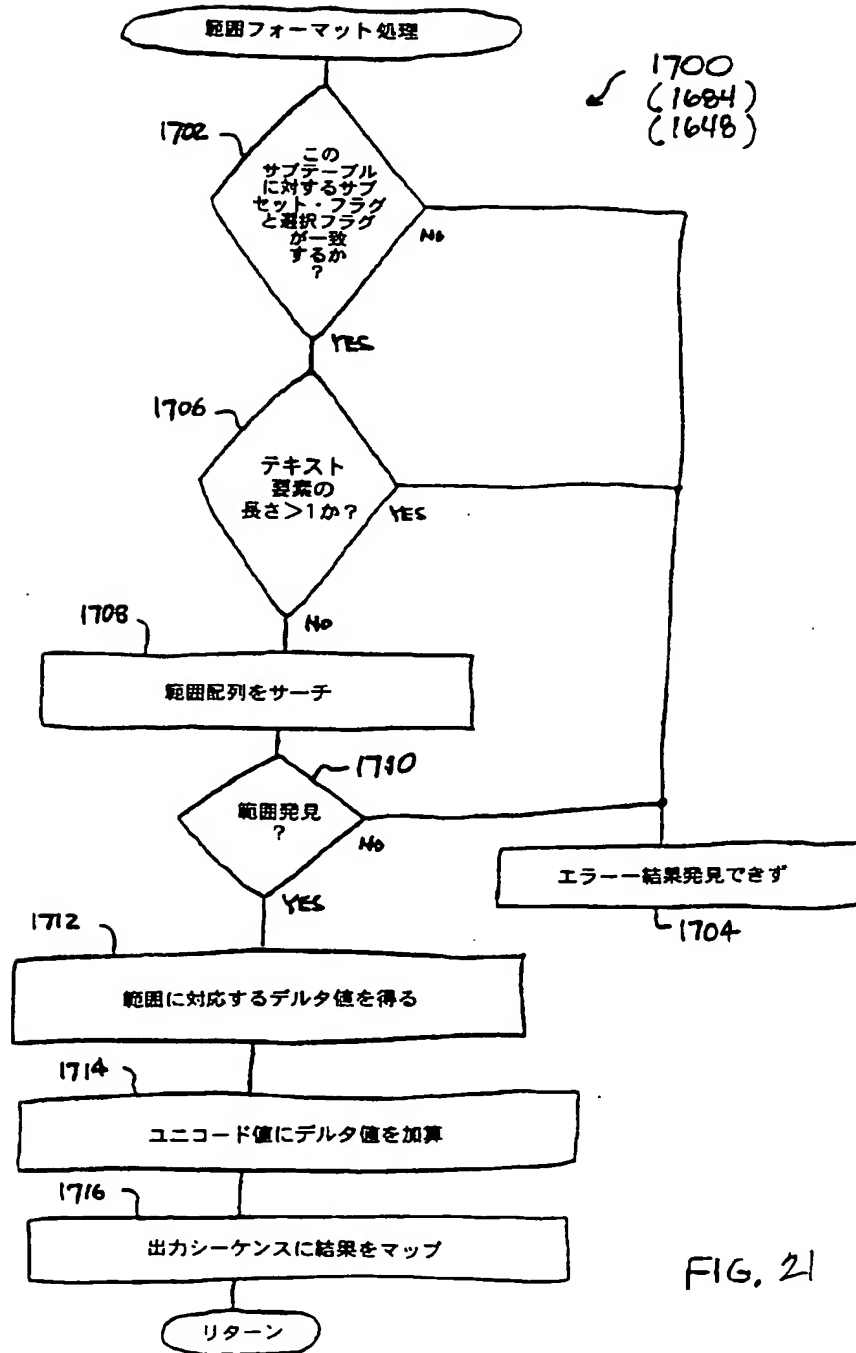


FIG. 21

【図22】

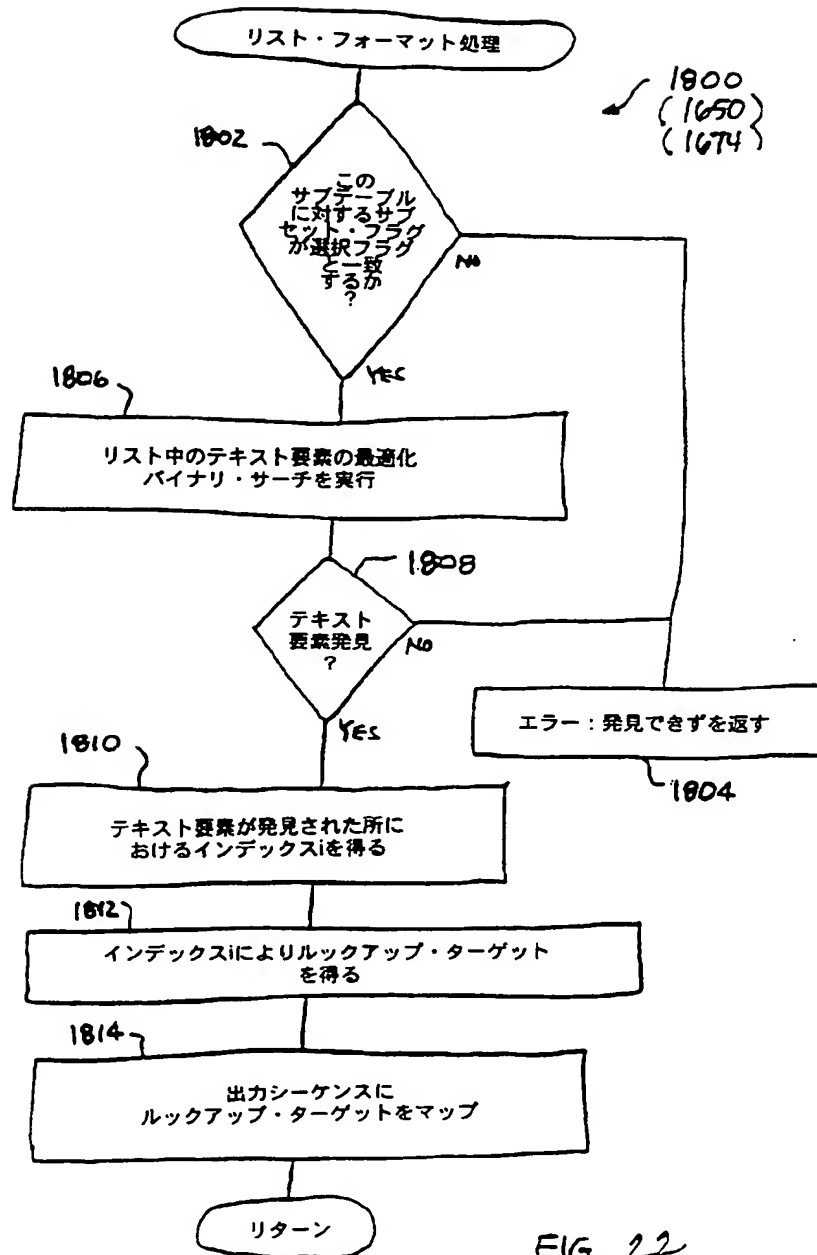
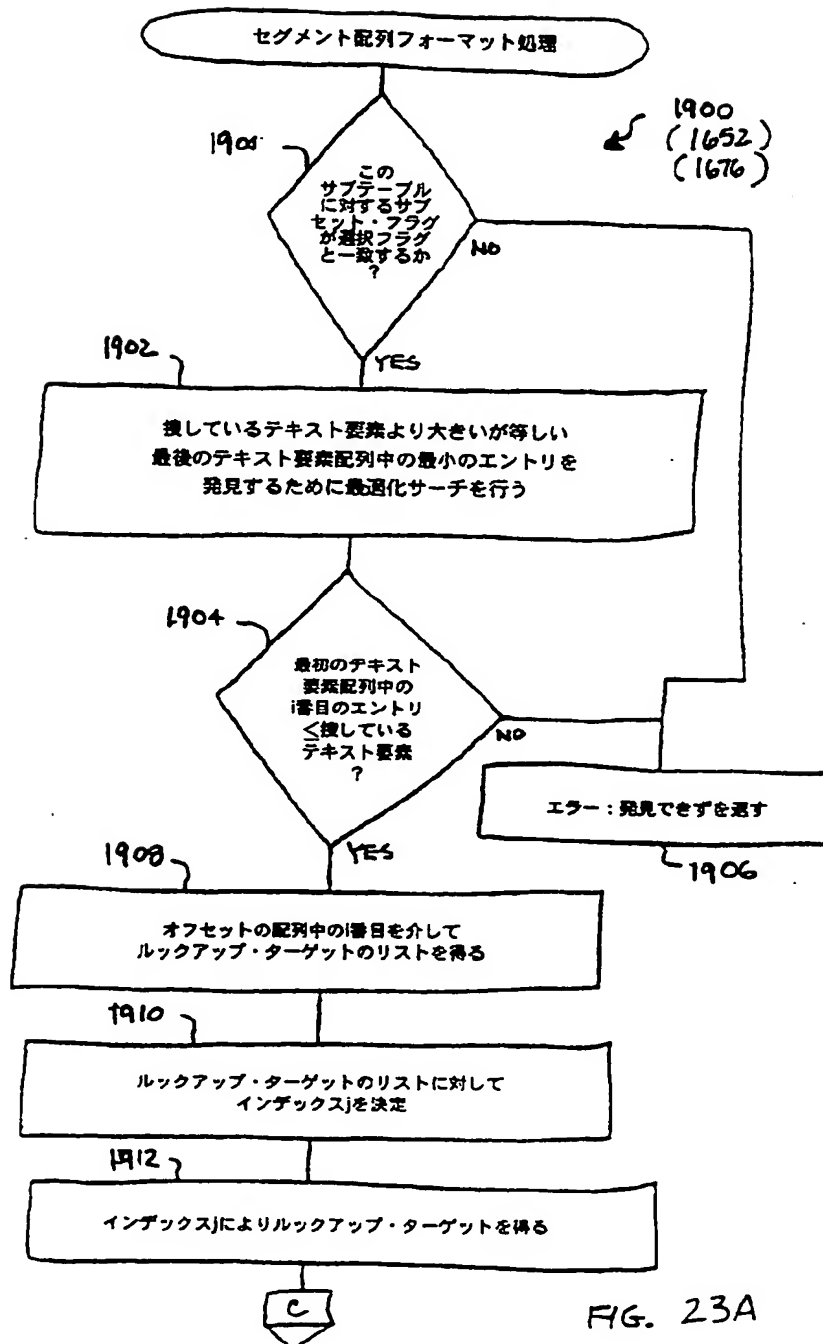


FIG. 22

【図23】



【図23】

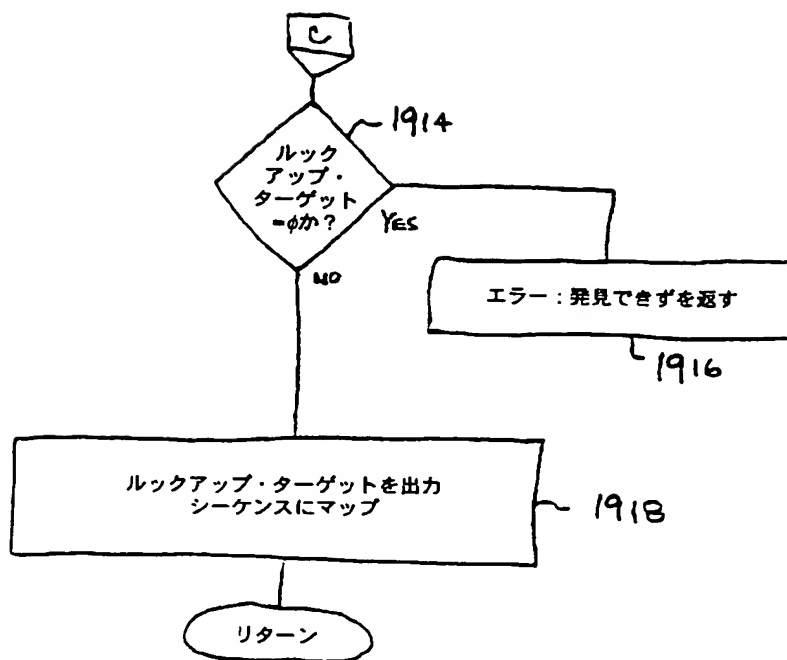
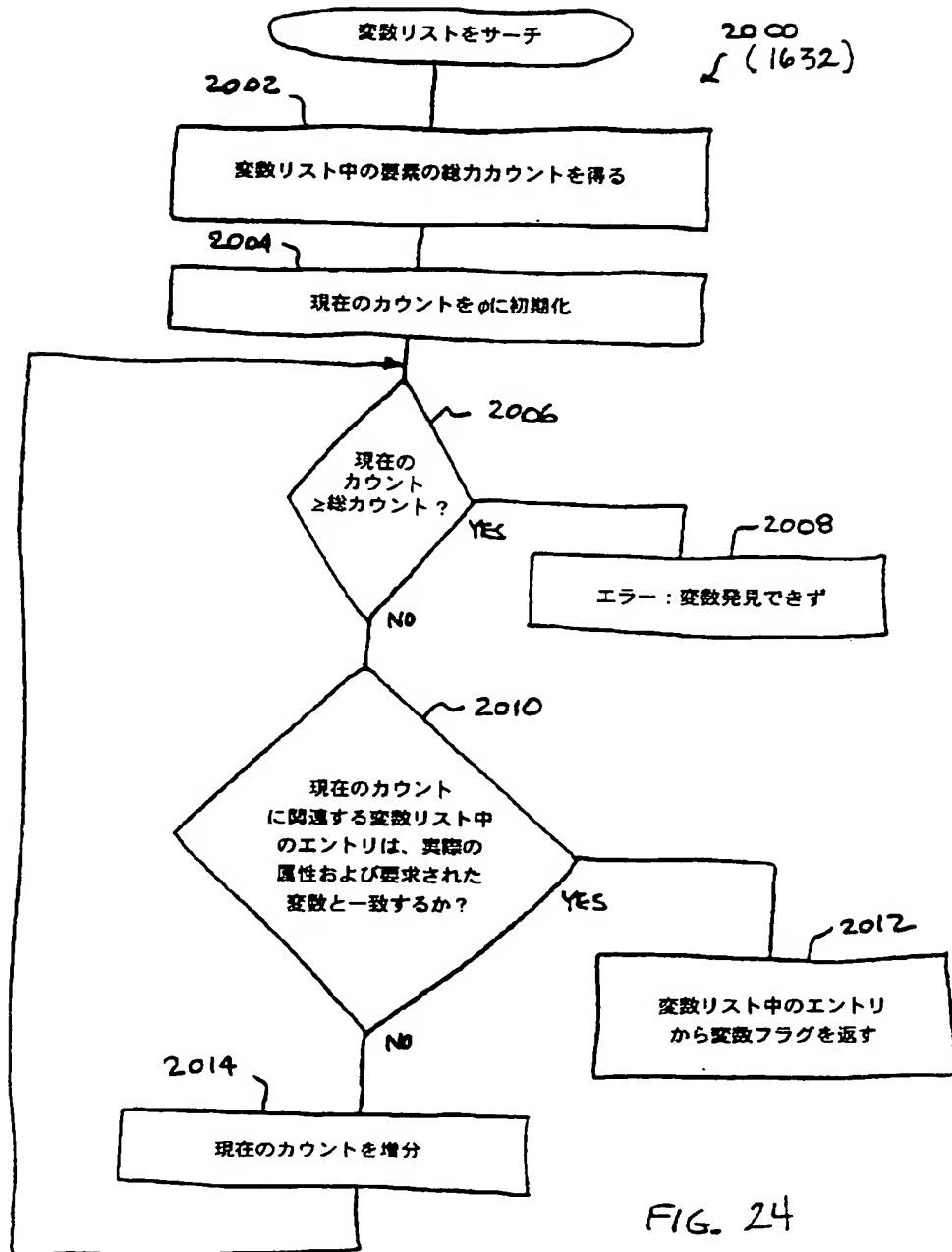


FIG. 23B

【図 24】



【図25】

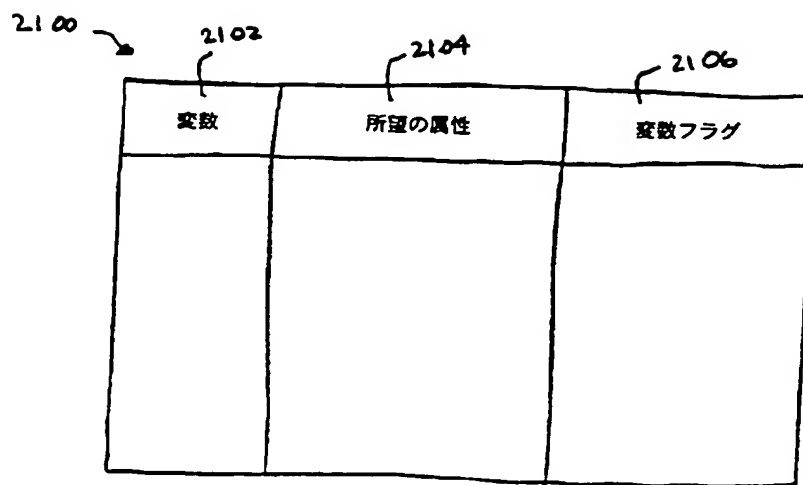


FIG. 25A

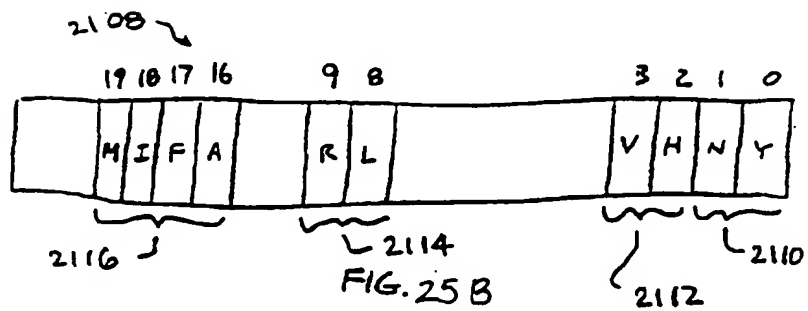


FIG. 25B

【図26】

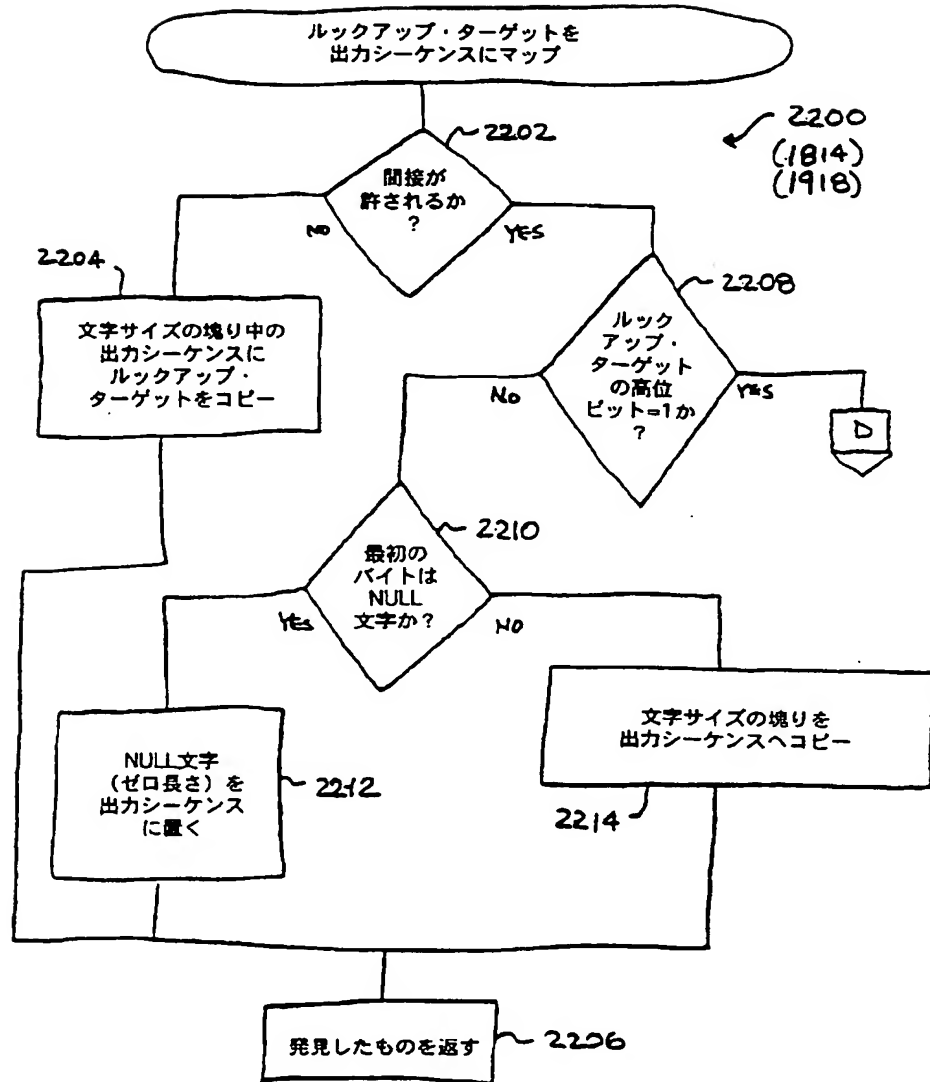


FIG. 26A

【図26】

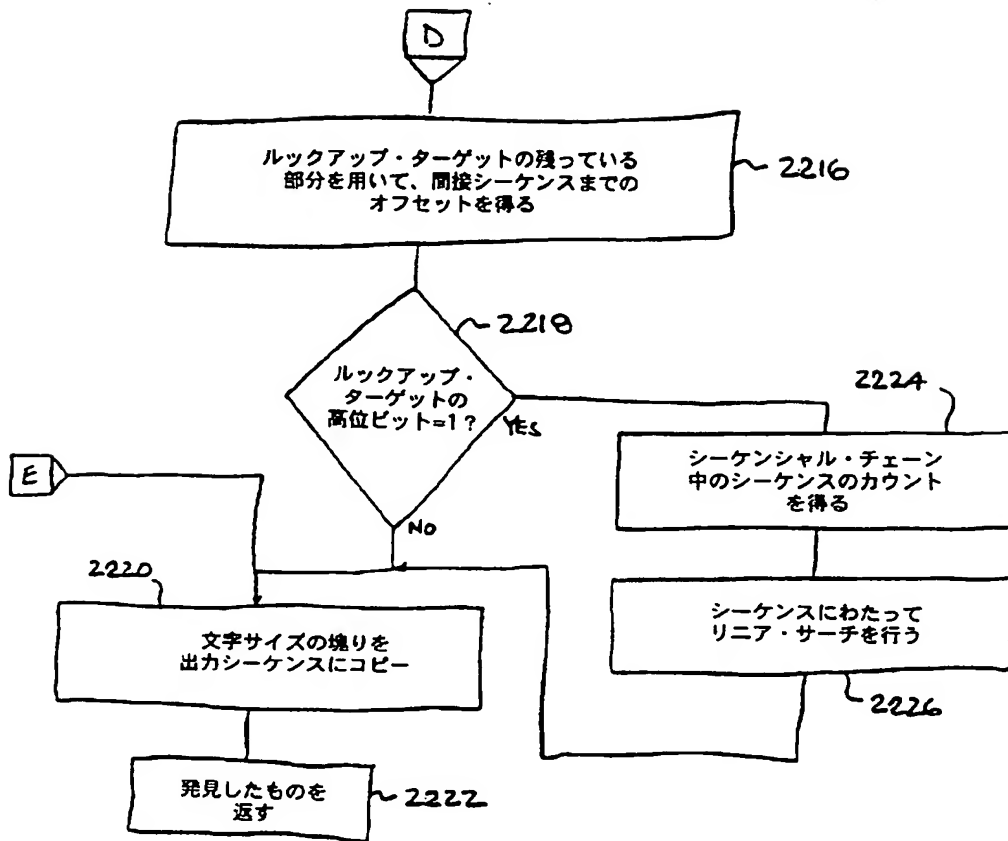


FIG. 26B

【図26】

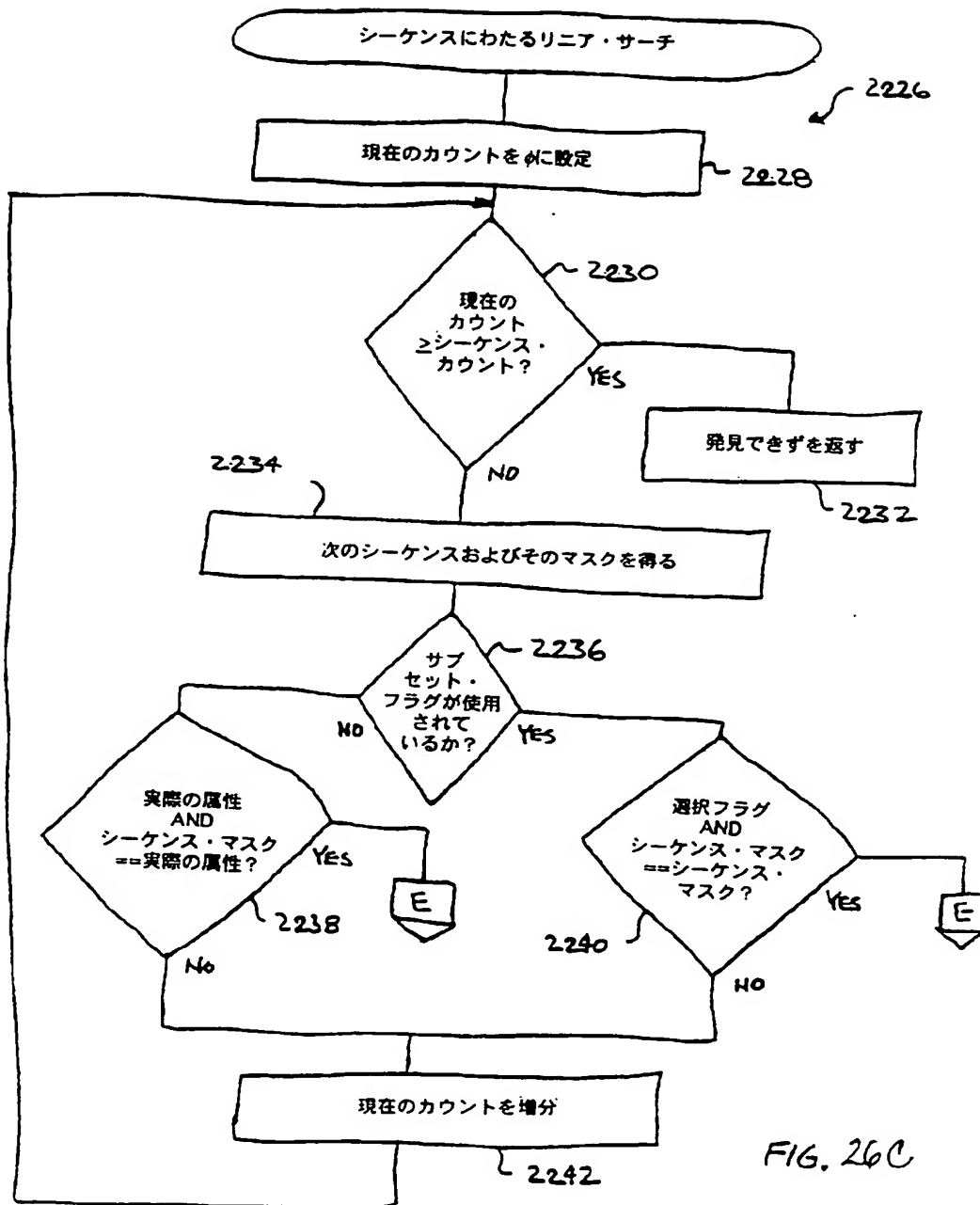
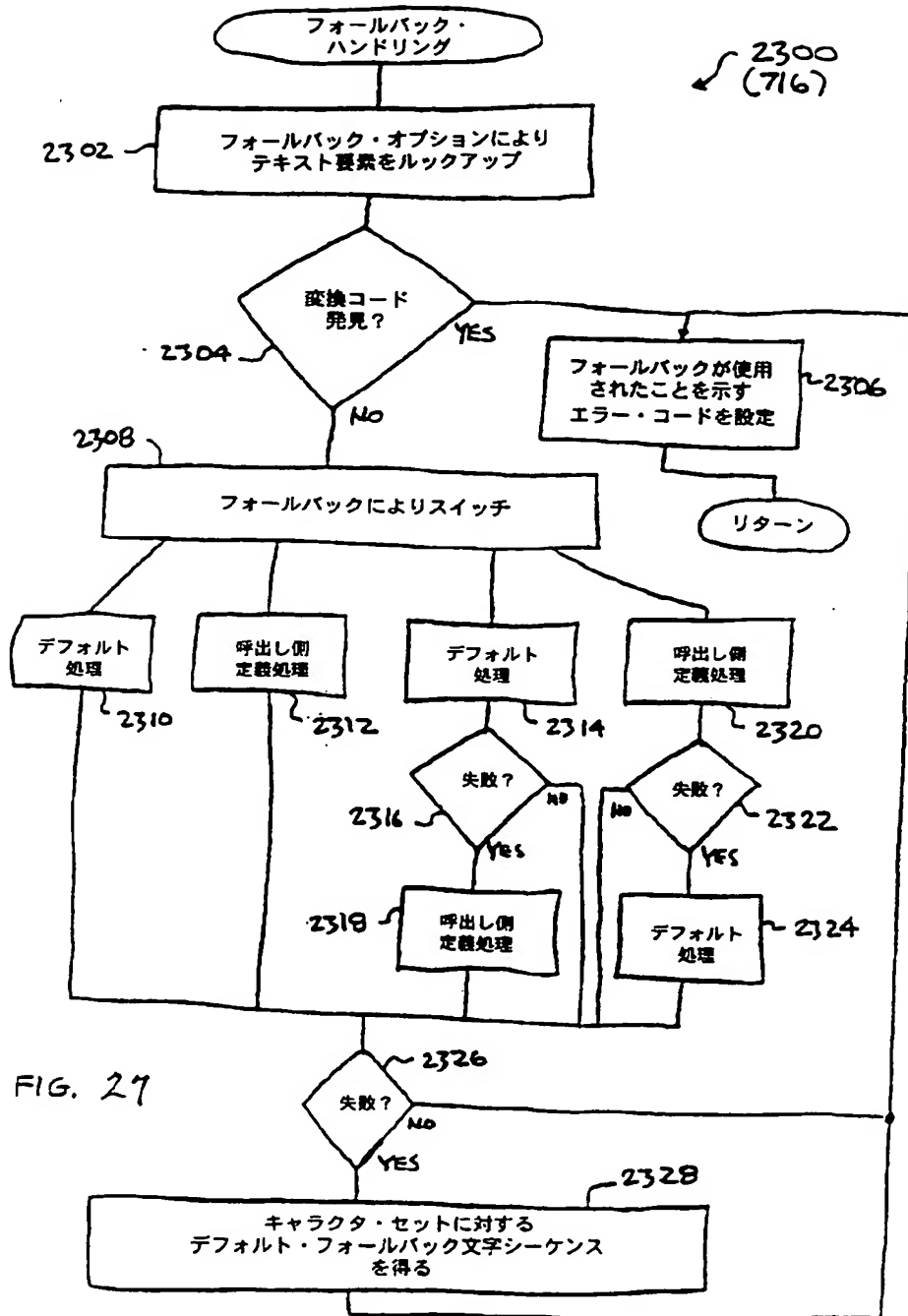


FIG. 26C

【図27】



【図28】

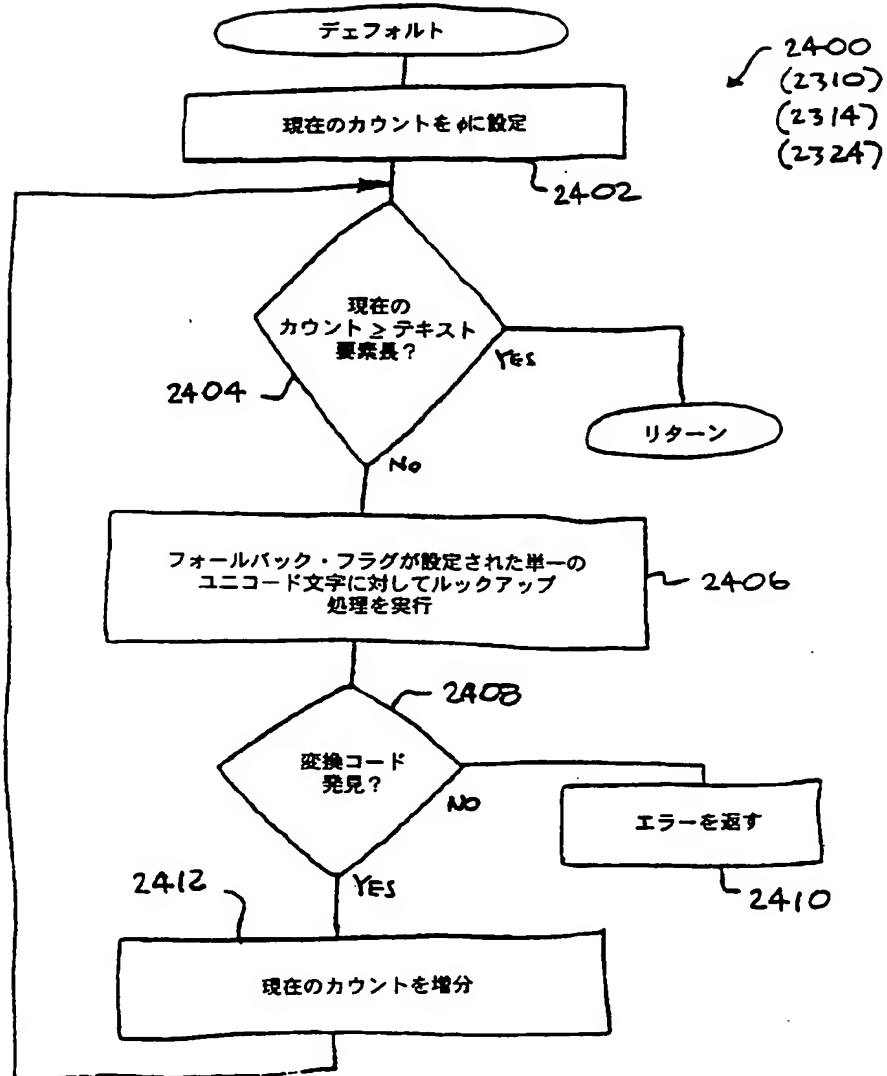


FIG. 28

【図29】

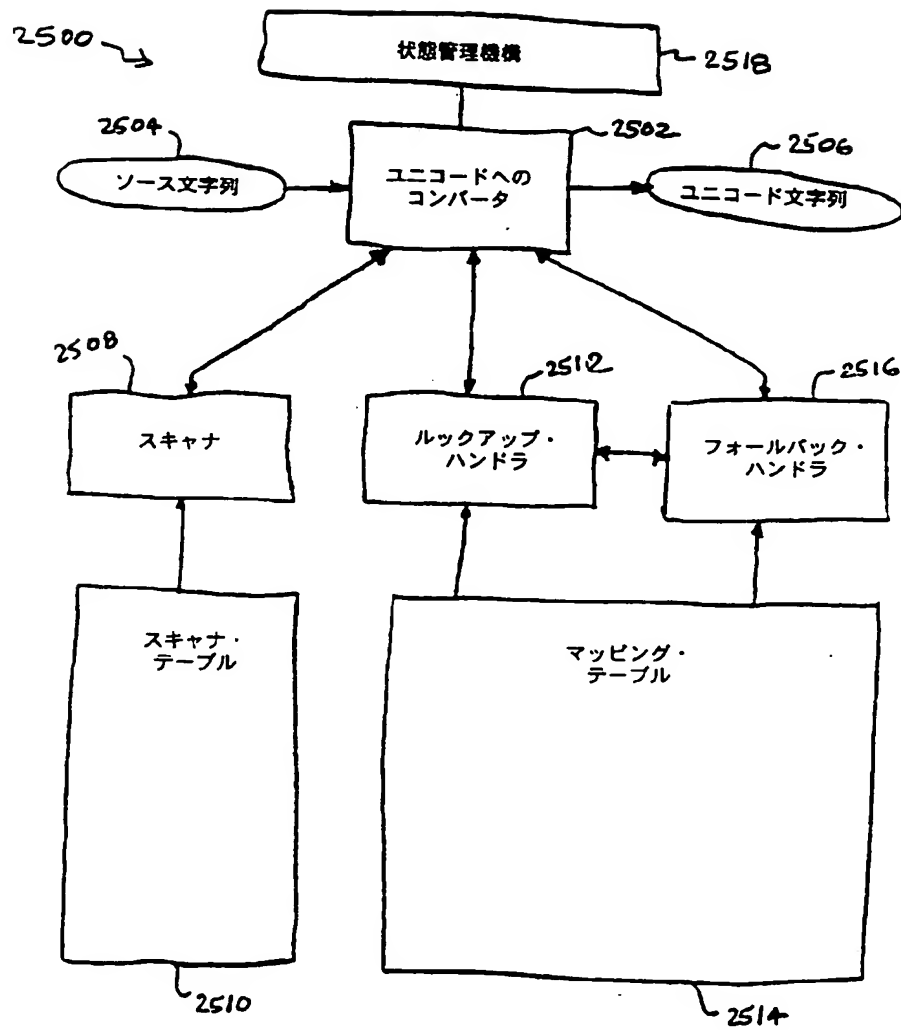


FIG. 29

【国際調査報告】

INTERNATIONAL SEARCH REPORT		International Application No. PCT/US 96/14667
A. CLASSIFICATION OF SUBJECT MATTER IPC 6 G06F17/22		
According to International Patent Classification (IPC) or to both national classification and IPC		
B. FIELDS SEARCHED Minimum documentation searched (classification system followed by classification symbols) IPC 6 G06F		
Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched		
Electronic data base consulted during the international search (name of data base and, where practical, search terms used)		
C. DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US,A,5 309 358 (ANDREWS ET AL) 3 May 1994 see column 2, line 54 - column 3, line 32 --- -/--	1,4,11, 12,14, 21,24, 27,39, 46,50, 64,71, 75,85,88
<input checked="" type="checkbox"/> Further documents are listed in the continuation of box C. <input checked="" type="checkbox"/> Patent family members are listed in annex.		
* Special categories of cited documents: "A" document defining the general state of the art which is not considered to be of particular relevance "E" earlier document but published on or after the international filing date "L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified) "O" document referring to an oral disclosure, use, exhibition or other means "P" document published prior to the international filing date but later than the priority date claimed "T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention "X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone "Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combinations being obvious to a person skilled in the art "Z" document member of the same patent family		
Date of the actual completion of the international search 22 January 1997		Date of mailing of the international search report 30.01.97
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Telex 31 451 epo nl, Fax (+31-70) 340-3016		Authorized officer Pottiez, M

INTERNATIONAL SEARCH REPORT

Inter- national Application No.
PCT/US 96/14667

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT		
Category	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	<p>OBJECTS IN EUROPE, vol. 2, no. 4, July 1995, pages 14, 16/17, 22-24, XP000565458 M. DAVIS ET AL: "International Support in Applications and Systems Software: Part Two" see page 17, right-hand column, line 16 - line 37 see page 14, right-hand column, line 21 - line 30</p> <p>---</p>	<p>1-25, 27-48, 50-73, 75-90</p>
A	<p>PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS, LOS ANGELES, 4 - 7 November 1990, INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, pages 499-504, XP000215394 M. DAVIS ET AL: "Unicode" see page 504, left-hand column, paragraph 4 - paragraph 6</p> <p>---</p>	<p>1,14,21, 24,27, 39,46, 50,64, 71,75, 85,88</p>
A	<p>IBM TECHNICAL DISCLOSURE BULLETIN, vol. 37, no. 11, 1 November 1994, pages 11-14, XP000487148 "Validation of Double-Byte Character Sets Text for Prompting in a Language-Sensitive Editing System"</p> <p>see page 12, paragraph 1 see page 13, paragraph 3</p> <p>---</p>	<p>6-8,14, 16,17, 21,22, 24-26, 35,39, 40,42, 43,47, 55,56, 61,64, 65,67, 71,72,75</p>
A	<p>IBM TECHNICAL DISCLOSURE BULLETIN, vol. 35, no. 5, 1 October 1992, ARNONK, US, page 273 XP000312973 "Technique to Provide NLS Code Page Conversion with Roundtrip Fidelity" see the whole document</p> <p>-----</p>	

INTERNATIONAL SEARCH REPORT

international application No.

PCT/US 96/14667

Box I Observations where certain claims were found unsearchable (Continuation of item 1 of first sheet)

This International Search Report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☒ Claims Nos.: 26, 49, 74, 91, 92
because they relate to subject matter not required to be searched by this Authority, namely:
Computer readable medium containing program instructions-Presentation of Information (Rule 39.1(V))
2. ☐ Claims Nos.:
because they relate to parts of the International Application that do not comply with the prescribed requirements to such an extent that no meaningful International Search can be carried out, specifically:
3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box II Observations where unity of invention is lacking (Continuation of item 2 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

1. ☐ As all required additional search fees were timely paid by the applicant, this International Search Report covers all searchable claims.
2. ☐ As all searchable claims could be searched without effort justifying an additional fee, this Authority did not invite payment of any additional fee.
3. ☐ As only some of the required additional search fees were timely paid by the applicant, this International Search Report covers only those claims for which fees were paid, specifically claims Nos.:
4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this International Search Report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remarks on Protest

- ☐ The additional search fees were accompanied by the applicant's protest.
- ☐ No protest accompanied the payment of additional search fees.

INTERNATIONAL SEARCH REPORT

Information on patent family members

Int. Application No.

PCT/US 96/14667

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US-A-5309358	03-05-94	CN-A- 1075563	25-08-93
		JP-B- 2502021	29-05-96
		JP-A- 6083571	25-03-94

フロントページの続き

(31)優先権主張番号 08/527, 831

(32)優先日 1995年9月13日

(33)優先権主張国 米国 (US)

(31)優先権主張番号 08/527, 837

(32)優先日 1995年9月13日

(33)優先権主張国 米国 (US)

(81)指定国 EP(AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OA(BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG), AP(KE, LS, MW, SD, SZ, UG), UA(AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, US, UZ, VN

(72)発明者 マコンネル, ジョン, アイ.

アメリカ合衆国 94025 カリフォルニア
州 メンロパーク マッケンドリー ドラ
イブ 222

(72)発明者 タン, ユン-フォン, フランク

アメリカ合衆国 94082 カリフォルニア
州 サニーヴェイル イースト ホームス
テッド ロード 175 アpartment
ナンバー 8

(72)発明者 ダニエルズ, アンドリュー, エム.

アメリカ合衆国 94025 カリフォルニア
州 メンロパーク ハイト ストリート
123

【要約の続き】

からユニコード文字に変換したり、ユニコード文字から他の文字セットに変換したりするとき利用すると、特に便利である。変換される文字の方向（向き）を判断することにより、および／または文字のコンテキストを判断することにより、コード変換システムは判断または解決された文字の方向を利用してターゲット文字コード化への正しいマッピングが得られることを保証する。切り捨て処理手法は入力バッファに置かれているソース文字列の一部を切り捨てて、切り捨てられた部分が、ソース文字列の中の後続文字に影響されことなくターゲット文字コード化に変換されることを可能にするように作用する。